1-1-2000

Developing a hypertext-based electronic reference library for a public transportation agency

Chun Li
Iowa State University

Maintenance of a hypertext-based electronic reference library for a public transportation agency

by

Lifeng Li

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Civil Engineering (Construction Engineering and Management)

Program of Study Committee:
Russell C. Walters, Major Professor
Charles T. Jahren
Lee B. Honeycutt

Iowa State University
Ames, Iowa
2002

Graduate College

Iowa State University

This is to certify that the master's thesis of

Lifeng Li

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Performance of construction projects heavily depends on how well information is managed. Attracted by the many advantages that information technology (IT) may bring to construction information management, an electronic publication called Electronic Reference Library (ERL), which utilizes information storage and distribution methods enabled by Information Technologies, is jointly developed and maintained by Iowa Department of Transportation (IDOT) and Iowa State University (ISU) since 1998.

After initial development, a new version of the ERL is released every six months by IDOT. As this process will continue in the foreseeable future, a well defined working procedure with effective and efficient methods on ERL maintenance will significantly reduce cost and time in the future. Based on the review of early research work of the ERL project and available technologies, a few tools and methods are propose and developed for the ERL maintenance task. To speed up HTML document preparation and error checking in PDF files, a method of using *gawk* and *Perl* scripts to automate the preparation and error checking procedures is discussed and proposed in this thesis. To make the scripting easier, a few changes are suggested to be made to the ERL file storage structure and the HTML codes. Documentation of the working procedures and a Project Specific Web Site (PSWS) were used in the ERL project to improve communication and share of knowledge. A few methods of collecting users' feedback were also discussed because user input has always been a major impetus and source for improving the ERL.

Though only five documents are collected in the IDOT ERL currently, technologies reviewed in this thesis and methods proposed for ERL maintenance can be helpful for construction document management in a more generic context.

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

Determined by some features of the construction industry, such as the large volume of the products, the uniqueness of the products and fragmentation of the industry, performance of construction projects depend largely upon how efficiently and effectively information is collected, stored, retrieved, communicated and distributed. Traditionally, information is stored in paper-based documents, and communicated and distributed by means of phone calls, fax, letters, and meetings. Information Technologies (IT), specifically web-related technologies, which have become prosperous in the past decade, have brought new tools for information management. Solutions based on IT technologies and products have many advantages over traditional paper-based method in terms of costs, efficiency and accessibility. Publishing standard specifications and manuals on electronic media such as CD-ROM's and the Internet has been popularly adopted by Departments of Transportation (DOT) in many states of the United States in recent years. A research project concerning development and maintenance of one of such electronic publications called the Electronic Reference Library (ERL), which collects several standard contract documents published by Iowa Department of Transportation (IDOT), has been conducted jointly by Iowa DOT and Iowa State University since 1998.

After initial development, the ERL is to be updated very six months and new versions of ERL will be published every April and October. There is no end to these updates in the foreseeable future. Thus significant amount of cost can be saved throughout the life cycle of the ERL if efficient and effective methods for ERL maintenance can be found and documented for successive ERL maintainers.

The ERL project team faces a few challenges in the maintenance process:

1. As most ERL project team members have a civil or construction background, learning the IT tools and products constitutes a major challenge to the ERL team members.

2. Each update of the ERL must be finished within a restricted period of time to avoid influence on letting of public works. Thus an efficient and effective update method would be of great help for ERL maintenance.
3. The long life expectancy determines that turnover of project members would be an inevitable problem for the project. The issue of retaining knowledge of leaving team members and training of new members is of vital importance to the project performance.
4. Fragmentation is another issue which cannot be neglected. Project team members come from Iowa State University and Iowa DOT, and work at different locations. And the documents collected in the ERL are compiled by various offices within the DOT. Effective communication and collaboration among project members would significantly improve efficiency of ERL maintenance.

1.2 Objectives

The primary objective of this thesis is to address the challenges listed above and provide efficient and effective methods and tools for maintenance of the ERL.

Though the ERL project focuses on improving storage, distribution of some specific large volume contract documents, many of the tools, methods and lessons learned in this project is helpful for project document management in a more generic context.

1.3 Research Outline

The ERL development and maintenance project is a joint effort by Iowa DOT and Iowa State University dated back to 1998. Though an express research plan is not stated on any of the literatures available so far, a clear research outline exists. Stages in this project include:

- Understand users' needs
- Review existing e-publications produced by other DOT's
- Technology review
- Develop a Master Plan for the ERL
- ERL implementation

- ERL maintenance

1.4 Thesis Outline

Many graduate students and undergraduate students have joined and left the ERL project team since 1998. Many research tasks have been completed in earlier stages. To help readers to have a complete idea of the ERL research project and hence better understand the issues discussed in this thesis, a brief review of earlier research results will also be presented in the thesis.

This thesis will focus on:

1. Technology Review, which is a review of technologies that may be adopted in the ERL project.
2. Suggested ERL maintenance methods and tools, which include:
    a. Automating HTML and PDF file preparation and modification using *awk* and *Perl* scripts;
    b. Documentation of the ERL development and maintenance procedures;
    c. Using Project Specific Web Site for knowledge and information management in the ERL project;
    d. Methods of collecting users' feedback;
3. Improve Maintainability of the ERL:
    a. Using a standardized document file structure to improve maintainability of the ERL;
    b. Using More Structured HTML in the ERL.

CHAPTER 2. THE ELECTRONIC REFERENCE LIBRARY

2.1 Introduction of the ERL

Construction and maintenance of the public transportation infrastructure constitute an important portion of the Architecture/Engineering/Construction (AEC) market of the United States. Each year the Iowa Department of Transportation (IDOT) let out hundreds of contracts to keep the public road system of Iowa function properly for the many road travelers. To make sure these works are designed and constructed in a controlled manner, the IDOT publishes a series of standard contract documents, such as: *Standard Specifications for Highway and Bridge Construction*, *Supplemental Specifications*, *Standard Road Plans*, *Materials Instruction Memorandums*, and the *Construction Manual*, etc. Users of such documents include contractors, design professionals, and IDOT inspectors. Originally these documents were only available in hardcopies, which had been the only form of books for hundreds of years in human history. While using these thick, heavy hardcopies may not cause much inconvenience to office workers, the burden of carrying these books is a big headache for the IDOT inspectors who travel from job site to job site and need the books to verify that the job is being constructed according to the contract documents. The emergence of e-books, which use the web publishing and CD-ROM storage technologies, provides new solutions to solve this problem. Being an institute ready to embrace new technologies, IDOT began to explore the possibility of developing the ERL, a collection of electronic versions of hypertext formatted contract documents, with the help of researchers from Iowa State University (ISU) in 1998.

2.2 Review of Study Results from Earlier Researches

Research of ERL development at IDOT and ISU first started in 1998. Before the author of this thesis joined this project in January 2001, a number of issues had been studied and two test versions and a production version of the ERL had been released. Issues considered by earlier researchers include (Chun Li, 2000):

- Understand the desires of target users and develop requirements for the ERL design;
- Review electronic publications produced by other DOT's;
- Develop a master plan for ERL development;
- Implementation of the ERL;

Results of these studies will be briefly introduced in this section.

2.2.1 Understanding Users' Needs

One of the characteristics of the ERL is that it has a well defined target user group consisting of engineers, inspectors and contractors. Understanding users' needs and establishing design requirements for the ERL was considered the first step in this project. Two focus group meetings and a number of interviews with potential users were conducted from August 1998 to January 1999.

A number of important findings were identified through the meetings and interviews with regards to desired contents in the ERL (Table 2-1, Chun Li, 2000), and desired attributes in the ERL (Table 2-2, Chun Li, 2000).

Meanwhile, a few concerns were also identified regarding legal status of the ERL such as accuracy, authenticity, etc.

2.2.2 Review of e-publications produced by other DOT's

A few similar electronic publications had been published by other DOT's and almost all state DOT's publish their standard contract documents on their web site. Early researchers conducted a review of such e-publications and the online documents. The review focused on the following aspects of the subject e-publications and online documents (Chun Li, 2000):

- Functions provided by the e-publications, such as electronic forms, search engine, etc;
- Document format. File formats used by the DOT's include: HTML, PDF, Word, and CADD. Navigation architecture.

- Early researchers developed a site map for Washington DOT web site, based on which a site map was proposed for the beta version of the ERL. The site map was found to be a useful tool for hypertext application development.

Table 2-1 Desired Contents in the ERL

Standard Specifications	General Supplemental Specifications
Design Manuals	Road Standards
Materials Instruction Memorandums	Construction Manual
Telephone books	Letting dates
Iowa DOT programs	Trade association manuals
Job site posting requirements and posters	Davis Bacon wage rates
List of who is under contract for which jobs	County IM's
Bid item list	Average unit prices
CFR sections that apply	Iowa Code section that apply
Forms (fill out and send electronically)	Standard proposal notes
Urban Standard Specifications for Public Improvements	ASTM, AASHTO, MUTCD manuals – subject to copyright restrictions
DBE information as of issuance date	Equipments rental rates -- subject to copyright restrictions
Links to internet sites	

Table 2-2 Desired Attributes in the ERL

User-friendly	Good search engine
Easy to read	Printable
Hyperlinks between documents	Able to open multiple windows
Able to fill in and send form	Able to ensure authenticity
Will retrieve correct documents when letting date is input	Provide technical support
Electronic documents similar to paper documents	Make connections between ERL and SiteManager®

2.2.3 Master Plan for ERL Development

Then a master plan for ERL development was developed and proposed by early researchers. Issues considered in this stage include:

Desired Contents in the ERL

In addition to *Standard Specifications* and *General Supplemental Specifications*, a few documents were identified as "being updated periodically and used in design and construction of most transportation projects", and should be included in the ERL with first priority (Chun Li, 2000). These documents include:

- *Materials Instructional Memorandums*
- *Standard Road Plans*
- *Standard Bridge Plans*
- *Road Design Aids Manual*
- *Bridge Design Manual*
- *Construction Manual*

Contents listed in Table 2-3 are considered as "would add value to the ERL" and should be included in the ERL in the future (Chun Li, 2000).

Table 2-3 Suggested Content for Future Versions of the ERL

Telephone books: Iowa DOT Local jurisdiction officials Contractors Consultants	Links to other internet sites (users could link to the site directly to the CD if on-line)
	Requirements for job site posters and electronic version of posters that could be printed by user
Davis Bacon wage rates	Computer programs developed by IDOT for use by contractors, consultants, and IDOT employees
Historical units prices	Bid letting dates
Design aids published by trade associations	IDOT web site
Standard proposal notes	Applicable CFR and Iowa Code sections

Text Structure and Navigation

Three types of navigation structure were considered, hierarchical, non-linear, and mixed (Chun Li, 2000).

In a strict Hierarchical text, nodes are linked to form a strict hierarchy, in which a node can only access those nodes directly above and below it. In a none-linear text, nodes are connected to form a complex network based on a large number of referential links. Mixed text has a basic hierarchical structure with a number of referential links that allow users to jump across the branches of the hierarchy.

The mixed text approach is considered more suitable because of the organization and cross-references in the documents and is actually applied in the ERL.

Links

Hypertext links can be classified as two basic types: navigational and associative links. Navigational links connect main content with other sub-content and function as path-finding tools. They serve as backbone of a user interface. Associative links offer parenthetical material, footnotes, digressions, or parallel themes that the author believes will enrich the main content (Chun Li, 2000).

To avoid or minimize disorientation of the users and other adverse side-effects of using hyperlinks, two approaches of making associative links are discussed:

- Open the referred content in a new browser window;
- Use frames.

Frames

Early researchers decided to use frames in the ERL. A three-frame presentation was applied for most contract documents included in the ERL.

Distribution

Currently, electronic documents can be distributed in three ways: CD-ROM, Intranet, and Internet, all of which are to be used in the ERL project. Advantages and disadvantages

of each approach are discussed in early researches. DVD and handheld devices are also identified as possible solutions in the future.

Evolving Maintenance Model

The ERL maintenance process used by early researchers and a recommended maintenance process are documented in Chun Li's thesis (2000).

2.2.4 Implementation

Implementation of the ERL was divided into four stages: alpha version, beta version, pilot version, and the first mass published version. A hypertext application development process (Figure 2-1) proposed by Kukkonen (1994) was adopted as a guideline for implementation of the ERL. A few issues were considered as major concerns at this stage:

1. Document Format. Two file formats are used in the ERL: Hypertext Markup Language (HTML) or Portable Document Format (PDF). Textual documents are converted into HTML files. Documents with complex forms and plans are converted to PDF formats.
2. Preparation of electronic document files. The contract documents are originally compiled in Word/WordPerfect or CAD programs. Converting such documents to the desired format and inserting links and bookmarks in the converted files constitute significant portion of the development work.
3. Implement the desired attributes. A few measures were taken to implement the following attributes in the ERL:
    a. Usability
    b. Consistency
    c. Portability
    d. Flexibility
4. Managing Development of the ERL as a project. A few people were involved in the ERL project. Task assignment, specialization, communication, and coordination issues have to be taken care of to ensure performance of the project.

An alpha version and a beta version of the ERL were released before mass production. User feedbacks were collected and analyzed for improving the ERL.

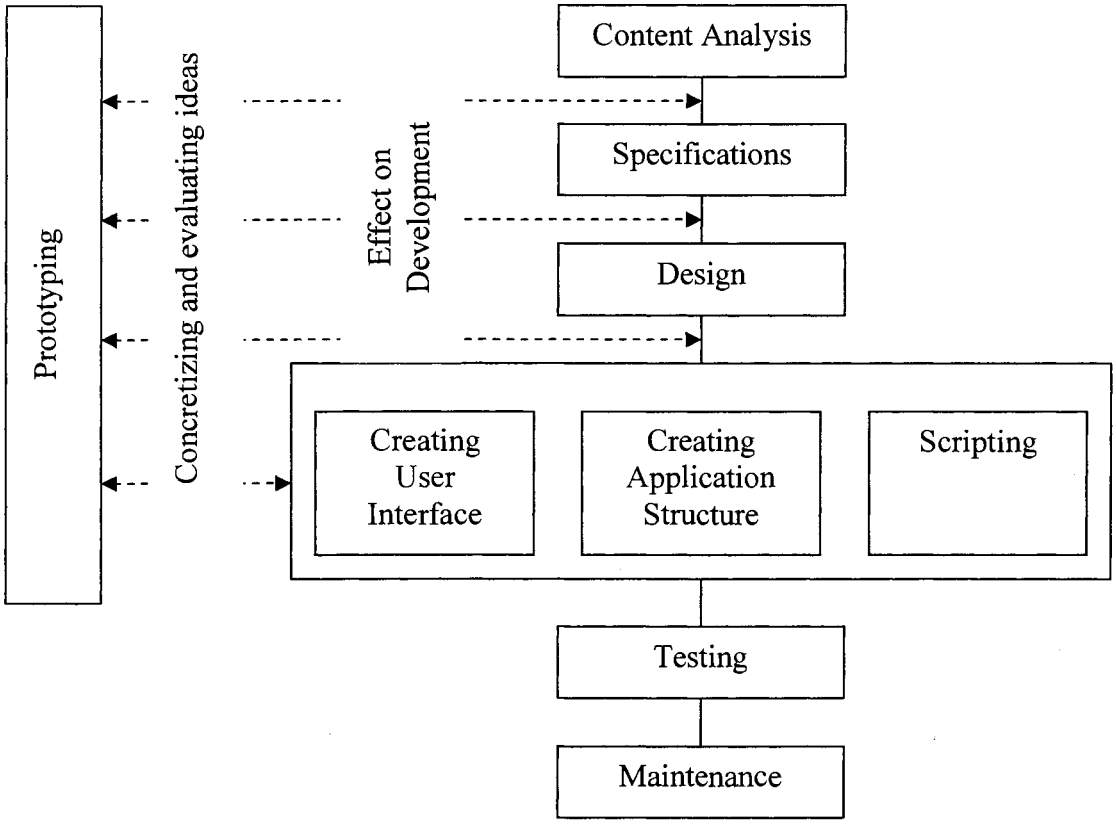


Figure 2-1 The Hypertext Application Development Process (Kukkonen, 1994)

CHAPTER 3. TECHNOLOGY REVIEW

Though demand for better information management tools is apparent, the adoption of IT applications in the construction industry is generally considered slow and poor. One of the major barriers is insufficient knowledge of available IT tools and products by construction professionals. Most project members having a civil or construction engineering background, the ERL project is faced with the same barrier too. To minimize the adverse effects of such insufficient knowledge, a review was conducted on available IT tools and products that may influence certain features or attributes of the ERL,

A second reason that a technology review is necessary now and should be conducted periodically in the future is because Information Technology, especially web technologies which are used in the ERL project, is an extremely active factor. New products from the IT industry may either bring new requirements or new solutions to the ERL.

As a result, a few factors are identified as being able or potentially able to bring improvements in the ERL and will be discussed in this chapter.

3.1 The Internet in Progress

The **Internet** is a global network of interconnected networks, connecting private, public, and university networks in one cohesive unit. It is a world wide computer network system with various types of hardware and software components, and huge amount of administrators, programmer, authors and users. Though the history of the Internet dates back to a few decades ago, the Internet is still young in that many new changes and improvements are happening or will happen. The Internet being one of the major distribution methods of the ERL, some of these changes are discussed here for their potentiality of bringing changes to the ERL.

Broader Bandwidth

CPU speed has been one of the foremost factors that determine the performance of a computer. Similarly, network bandwidth and speed of data transfer is one of the important factors that determine what type of services can be provided on the Internet, and the degree

of user's satisfaction with such services. WWW was once ironically interpreted as World Wide Wait. Slow speed is the primary reason of complaint for Web users. If the connection speed is lower than certain rate, some services, such audio and video chat or conference, will become useless at all.

Increasing the bandwidth of the Internet has been one of the prime objectives of researchers and equipment manufacturers. A revolution that will significantly increase the data and voice transmission speed is undergoing gradually. The method is using fiber optic cables in building new networks and replacing copper wires in existing networks. A few types of technologies are available for constructing Wide Area Networks (WAN) and Local Area Network (LAN). In a Local Area Network which uses Ethernet technology, CAT 5 copper wire cables have an upper bandwidth limit about 100M bps, while fiber optic cables have a bandwidth of 10G bps. Gigabit Ethernet using fiber optic has become an industry standard for years. 10-Gigabit Ethernet standard is nearing to completion (10 Gigabit Ethernet Alliance, 2002). Researchers are aiming at 40 Gigabit Ethernet or even 160 Gigabit for the next generation of Ethernet. Fiber optic cables are becoming more and more popularly used in constructing networks for new buildings.

Methods of "last mile" connection to end users are also gradually changing. ISDN, ADSL, and cable connection are gradually replacing 56k modems in more and more households. Community networks using LAN technology to connect common households are becoming wide spread in densely populated cities.

Due to many reasons, the upgrading of current networks to broader bandwidths takes time. But one thing is for sure, the transmission speed of data along the Internet will continuously increase at a firm pace for the next a few decades.

Wireless connection

Wireless connection is another favorable trend that is happening to Internet users. Two types of upcoming technologies have promised us many advantages that will be brought about by wireless connection. One of them is Wireless Local Area Network (Wireless LAN or WLAN), defined by IEEE 802.11 task group. The other is Third Generation of mobile

telecommunications (3G), which comes from mobile telecommunication equipment providers.

A wireless LAN (WLAN) is a flexible data communication system implemented as an extension to, or as an alternative for, a wired LAN within a building or campus. Using electronic waves, WLANs transmit and received data over the air, minimizing the need for wired connections. Benefits of WLANs include:

- **Mobility improves productivity and service** – Wireless LAN system can provide LAN users with access to real-time information anywhere in their organization. This mobility supports productivity and service opportunities not possible with wired networks.
- **Installation speed and simplicity** – Installing a Wireless LAN system can be fast and easy and can eliminate the need to pull cable through walls and ceilings.
- **Installation Flexibility** – Wireless technology allows the network to go where wires cannot go.
- **Reduced Cost-Of-Ownership** – while the initial investment required for wireless LAN hardware can be higher than the cost of wired LAN hardware, overall installation expenses and life-cycle costs can be significantly lower. Long-term cost benefits are greatest in dynamic environments requiring frequent moves, add, and changes.
- **Scalability** – Wireless LAN systems can be configured in a variety of topologies to meet the needs of specific applications and installations. Configurations are easily changed and range from independent networks suitable for a small number of users to full infrastructure networks of thousands of users that allow roaming over a broad area.

In a typical WLAN configuration, a transmitter/receiver (transceiver) device, called an access point, connects to the wired network from a fixed location using standard Ethernet cable. End users access the WLAN through wireless LAN adapters, which are implemented as PC cards in notebook computers, or use ISA or PCI adapters in desktop computers, or fully integrated devices within handheld computers. Typical throughput of WLAN ranges between 1 and 11 Mbps, which is comparable to current 10 Base-T network connections.

WLAN devices are already commercially available with some functionality, such as security, yet to be improved. (Introduction to Wireless LANs, WLANA)

Another existing method of connecting to the Internet wirelessly is through cellular phones. The problem with current mobile telecommunication devices is the limited bandwidth of about 9.6 Kbps. One of the major improvements promised by 3G is data rate, which will depend upon the environment the call is being made in (GSM World, 2002):

- High Mobility

144 kbps for rural outdoor mobile use. This data rate is available for environments in which the 3G user is traveling more than 120 kilometers per hour in outdoor environments. Let us hope that the 3G user is in a train and not driving along and trying to use their 3G terminal at such speeds.

- Full Mobility

384 kbps for pedestrian users traveling less than 120 kilometers per hour in urban outdoor environment.

- Limited Mobility

At least 2 Mbps with low mobility (less than 10 kilometer per hour) in stationary indoor and short range out door environments.

While many people describe Internet access as ubiquitous, it is not today. We have difficulties in finding an Internet port on trains, or airport terminals. These wireless technologies, combined with the mobile personal computing devices, such as notebook computers and PDA's, will definitely contribute to make Internet access truly ubiquitous.

Converged Network

We use a number of networks, such as wired and wireless telephone networks, computer networks, and cable TV networks, for our entertaining, communication, and learning purposes in our everyday life. As all these networks deliver digital signals and the capacity of the Internet is almost unlimited due to application of fiber optic cables, why bother use so many different separate networks? Based on this idea, the IT/telecommunication industry has spent significant effort in constructing a converged network on the basis of the widespread Internet. The converged network will be able to

provide voice, video and data services at the same time. Some commercialized solution is already available for enterprise users, while truly converged network that can fully integrate functions of the wired/wireless telephone network and the Cable TV network will remain an idea for some time.

Powerful and Versatile Personal Computing Devices

Improvement in computation and storage capacity of personal computers has been one of the staples that supports the rapid growth of the IT industry for the past a few decades. A perfect illustration of this trend is the renowned "Moore's Law". CPU speed has been increasing continuously, while unit prices for RAM and hard drives keep dropping. The direct impact of this trend on the Internet is that Servers on the net can respond more quickly to users' request and provide more information at lower costs. Cheaper hardware devices combined with the many free but reliable software packages, such as Linux, is making IT adoption much more financially justifiable to many enterprises.

Another emerging trend with personal computing devices is that they are going to be more portable, and more versatile. Laptop computers have been considered a high-end product aiming at business people who have to travel a lot. But the significant price drop that happened in recent years is gradually changing the concept. More and more private users are replacing their old and cumbersome desktops with portable and fashionable mobile computers. The upcoming trend of wireless network connection, which practically improves the mobility of laptop computers, will obviously contribute to the further popularity of laptop computers. Right now, mobile phones, PDA's (Personal Digital Assistant), and laptop computers are three different types of personal business equipment. This is changing as a result of the network convergence as described above. Computing and documenting functions are added to mobile phones. PDA's can be used to connect to the Internet, and can also be used to play multimedia files. Built-in wireless network cards are included in new laptop computer models. Sometime in the future, we may bring only one small machine with us when we travel. With it we can make phone calls, receive and send emails, listen to music, browse the Internet or corporate Intranet, and maybe even watch TV. This might sound too optimistic. At least it can be partially realized within the next a few years.

Deeper and Broader Application

Many usages have been found for the Internet, yet more are to be discovered or implemented. Many applications are mature and have been widely accepted, such as e-mail, ftp, web browsing, online shopping, discussion board, etc. Many more are still at the testing or spreading stages. Currently, applications of such kind include VOD (Video on Demand), Electronic Publishing / Digital Library, E-Learning and Education, E-Government, etc. The development of the Internet Infrastructure and related equipment and devices, is continuously providing new possibilities of change and improvement in many traditional industrial and academic areas. In many occasions, it is the task of professionals of such areas to explore and implement such possibilities, of course, with the help of IT professionals. Although we are not sure what will happen and when it will happen, the Internet is very likely to bring more changes to our life that it did.

Summary

The Internet is being changed by its participants. As one of the major approaches of ERL distribution, these changes in the Internet may bring new requirements to the ERL as well. Such changes might include:

- Shift in the major change of ERL distribution.
- Developing ERL's for new personal computing devices.
- New formats in the ERL, such as video and audio instructions, etc.

3.2 The Web as a Project Information Management Instrument

Many services are available on the Internet to meet users' needs. Among these, the World Wide Web is the one that causes most traffic on the Internet (Hobbes' Internet Timeline, http://www.zakon.org/robert/internet/timeline/).

Lemay (1997) summarized the following characteristics for the World Wide Web, which best illustrate why the WWW has achieved so great success and popularity in only a few years.

- The Web is a hypertext information system

- The Web is graphical and easy to navigate
- The Web is cross platform
- The Web is distributed
- The Web is dynamic
- Web browsers can access many forms of internet information
- The Web is interactive

These characteristics make the Web an excellent choice for construction project information management. Many research institutes and enterprises have conducted researches to explore effective solutions in applying web technologies in construction information management. Benefits of using the web as a project management tool also attracted interest from owners. To some, providing and managing a digital project information system is becoming a part of the standard scope of construction management services (3D/International, 2001). An integrated information management system based on web-database integration was proposed by Varney in 1997. New features of web and database support have been added to traditionally PC-based project management software packages such as Primavera Project Planner Enterprise Edition (P3), Meridian Prolog Manager, and Microsoft Project.

To most project participants, a web based project information system, normally referred as Project Specific Web Site (PSWS), can serve the following functions:

- Knowledge base

  The term Knowledge Management (KM) is becoming popularly used in recent years, which refers to the process that helps organizations identify, select, organize, disseminate, and transfer important information and expertise that are part of the organizational memory that typically resides within the organization in an unstructured manner (Turban, E., Aronson, J. E., 2001). Traditionally, knowledge is stored in the minds of individual members. The organization may easily get into trouble when the person leaves or is unavailable. The concept of KM advocates a sharing environment in which employees contribute their own knowledge to a knowledge base from which others may learn what they need. A PSWS makes an excellent platform for sharing knowledge relevant to the project.

- Interface to a centralized corporate database

  As a key element to Management Information Systems (MIS), a centralized database is gaining more and more importance in the construction industry. Estimators, schedulers, and project managers need historical data to predict future construction cost and time. Data about activities in the project at hand is needed to decide what to do next. By using a PSWS as the interface to a centralized database, project members can retrieve any information they need with a few strokes on the key board. They can also update the database with the latest information about certain tasks they have just completed.

- Information center

  The interactive property of a PSWS enables it to serve as a real time information center for project members.

- An integrated working environment

  Traditionally, construction engineers use different programs to complete certain tasks. A properly designed PSWS may allow project participants finish many tasks, such as materials procurement and management, document control, cost control, progress control, etc, within the same working environment of a PSWS.

The PSWS concept is related to the ERL project in two aspects: an ERL can be used as a part of a PSWS; and many functions of a PSWS is needed in the ERL development and maintenance process.

3.3 HTML 4 and CSS

3.3.1 HTML 4

With a history of only about ten years, the WWW is still a young product. Many of its components are continuously experiencing improvements. Among these is the Hypertext Markup Language (HTML), the most fundamental authoring tool for web publishing.

According to *HTML 4.01 Specification,* the latest recommendation by the World Wide Web Consortium (W3C), HTML is "the publishing language of the World Wide Web," and

"a kind of mother tongue that all computers may potentially understand." HTML is designed to serve such functions as:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

HTML was originally developed by Tim Berners-Lee in early 1990's and is currently maintained at W3C. Through the years a few versions of HTML specification have become RFC's or W3C recommendations. The latest version of HTML specification is 4.01, a subversion of HTML 4. HTML 4 extends HTML with mechanisms for style sheets, scripting, frames, embedding objects, improved support for right to left and mixed direction text, richer tables, and enhancements to forms, offering improved accessibility for people with disabilities (HTML 4.01 Specification).

3.3.2 Cascading Style Sheet (CSS):

One of the trends that is happening to HTML is that designers of HTML are trying to separate the structure and presentation functions of HTML. As HTML matures, more and more of its presentational elements and attributes are being replaced by other mechanisms, in particular style sheets (HTML 4.01 Specification). Style sheet is a set of statements that specify the presentation of a document.

Including style sheets in HTML is considered a breakthrough in HTML and web design. It is aimed to improve the presentational capability of HTML and intended by the HTML 4.0 specification authors to replace sidestep methods used in Web page design, which include:

- Using proprietary HTML extensions
- Converting text into images
- Using images for white space control
- Use of tables for page layout
- Writing a program instead of using HTML

Beyond these, there also many obvious advantages of using style sheets in HTML document and these advantages are especially important for web-sites or e-publications of large volumes. The advantages include:

- More structured HTML codes;
- Smaller HTML files;
- Centralized control over the appearance of the web pages, which mean more ease and convenience in modification;
- Consistent look throughout the Web site or e-publication.

Cascading Style Sheet (CSS) is a style sheet language recommended by W3C to work together with HTML. To understand how we can integrate CSS in a HTML document and the advantages of using CSS, we can compare the following two sets of HTML codes, which will appear the same through a web browser.

HTML code set 1:

```
<HTML>
  <HEAD>
  <TITLE>Bach's home page</TITLE>
  </HEAD>
  <BODY>
    <H1><font color="blue" face="arial">Bach's home page</font></H1>
    <P><font color="red">Johann Sebastian Bach was a prolific
composer.</font></P>
  </BODY>
</HTML>
```

HTML code set 2:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
  <TITLE>Bach's home page</TITLE>
  <STYLE type="text/css">
    BODY { color: red }
    H1 { color: blue;
         font-family: Arial
         }
  </STYLE>
```

```
  </HEAD>
  <BODY>
    <H1>Bach's home page</H1>
    <P>Johann Sebastian Bach was a prolific composer.
  </BODY>
</HTML>
```

The first set of codes uses the "font" element to specify font colors. Instead of using the "font" element, which is deprecated element HTML 4.01 and may become obsolete in future versions of HTML, the second example uses CSS rules to format the content. A CSS rule consists of two main parts: selector ('H1') and declaration ('color: blue'). The declaration has two parts: property ('color') and value ('blue').

Besides including the CSS codes in the HTML document as shown in example 2 above, HTML 4.0 also allows web authors to use external style sheets. Thus the same content of example 2 can be rewritten in the following way:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
  <TITLE>Bach's home page</TITLE>
  <LINK rel="stylesheet" href="bach.css" type="text/css">
  </HEAD>
  <BODY>
    <H1>Bach's home page</H1>
    <P>Johann Sebastian Bach was a prolific composer.
  </BODY>
</HTML>
```

The "bach.css" file refered in the above HTML code is a simple text file with the following contents:

```
BODY { color: red }
H1 { color: blue;
font-family: Arial
    }
```

Discussions on how to use HTML 4 and CSS to improve structure and maintainability of HTML files in the ERL will be discussed in Chapter 5.

CHAPTER 4. ERL MAINTENANCE TOOLS AND METHODS

Update and maintenance of ERL is a continuous effort. Every six months, a new revision of ERL, which contains new documents or changes to existing construction documents, will be released by IDOT. Constrained by the limited resources, the process of updating and maintaining the ERL is always a challenge to the project team. To reduce the difficulty and quantity of work, a clear objective was set up for this phase of the ERL project – to improve the maintainability of the ERL and provide efficient and effective maintenance tools and methods for later maintainers.

After carefully reviewing the ERL development and maintenance process and available technologies, a few measures were proposed and implemented. They include:

- Automating the Document Conversion Process;
- Documenting the ERL update and maintenance process;
- Using a Project Specific Web Site (PSWS) for the ERL project
- Understanding Users' Changing Needs.

4.1 Automating the HTML File Preparation and Modification Process

HTML File Development Process

The process of creating document files (HTML or PDF) can be summarized into a flowchart shown in Figure 4-1.

For the "Insert links and bookmarks in the HTML files" part of work, links and bookmarks were added one by one using FrontPage. For a normal document like the *Construction Manual* or *Standard Specifications*, it normally takes a person about 10 - 15 working days to finish this task. If the updater is involved in other works, the job could take more than a month to finish. Because of the tediousness of the work, errors are guaranteed to happen.

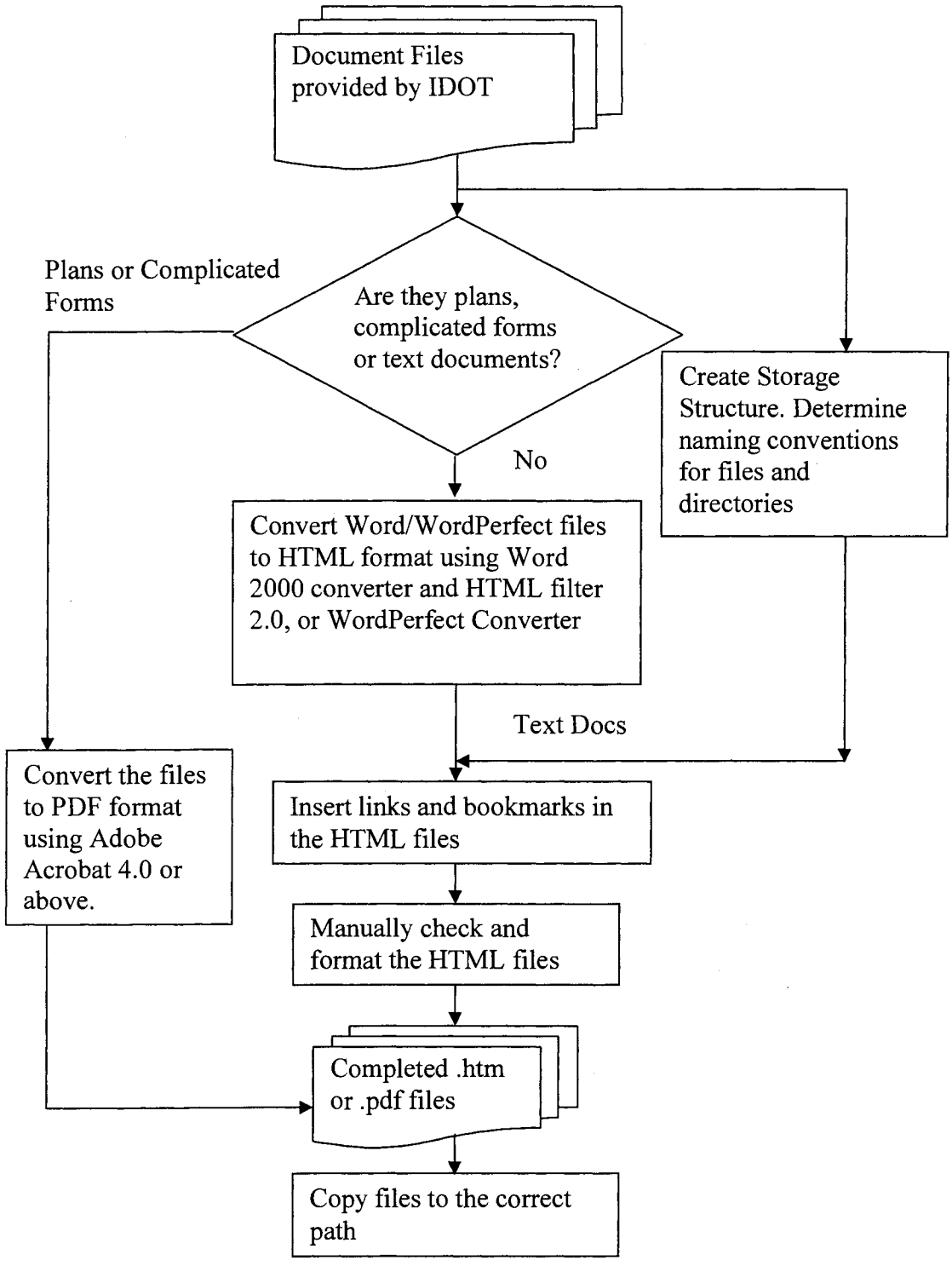


Figure 4-1Process of Preparing the Document Files

Typical errors found in the past include:

- Using absolute paths[1] instead of relative ones[2];
- Unlinked text;
- Broken links;
- Links refer to a wrong file.

Thus, another one or two weeks is needed to fix errors in these files. Restricted by the limited number of people directly involved in ERL development and maintenance, creating the document files by this way typically leads to postponement of the ERL release date.

Fortunately two convenient scripting utilities were identified and introduced into ERL development and maintenance for their capability of manipulating contents of text and binary files. These two utilities are *gawk* and *Perl*.

Gawk and *Perl*

The term *awk* refers to a particular program originally used in the UNIX operating system, and also to the language used to tell this program what to do. The term *gawk* refers to the GNU (http://www.gnu.org) implementation of *awk*. Many computer users frequently need to make changes to text files wherever certain patterns appear, or extract data from parts of certain lines while discarding the rest. This type of work is easier with *awk*. And the above-described editing work with ERL happens to fall into such a category.

Perl stands for *Practical Extraction and Report Language*. It is a utility program originally designed for manipulating text files and performing some other functions in the UNIX environment by Larry Wall. *Perl* is used here in addition to gawk because *Perl* can be used to manipulate binary files such as PDF files, another important file format used in ERL. Because of the powerful functions provided by *Perl*, it is widely used for CGI (Common Gateway Interface, a type of server side scripting used in web servers) programming.

Awk is an excellent tool to manipulate text files because it is easy to learn and simple to use. *Effective Awk Programming – A User's Guide for GNU Awk* by Anold D. Robins is an

---

[1] An absolute path refers to the name of the hard drive to locate the files, i.e. "C:\My Document\index.htm".

[2] Relative path refers to the referred file by using the relative location of the referring file itself, i.e. "../content/2001.htm" ("../" means go up one level of directory).

excellent source for learning how to use *awk* and can be accessed at http://www.gnu.org/manual/gawk-3.0.3/gawk.html. A brief introduction of how to use *awk* will be repeated here before we get into the introduction of the *gawk* script written for processing ERL HTML files.

Awk was originally a utility program for UNIX. Now it is available for Microsoft Windows based systems and many other operating systems. The basic function of *awk* is to search files for lines (or other units of text) that contain certain patterns. When a line matches one of the patterns, *awk* performs specified actions on that line. A pattern and an action together constitute a rule. An *awk* program can be made of one rule or a collection of rules. Therefore, an *awk* program looks like this:

```
pattern { action }
pattern { action }
...
```

Basically *awk* can be run in two ways. If the program is short, it is convenient to include it in the command that runs *awk*, like this:

```
$ awk `program' input-file1 input-file2
```

where $ is the UNIX prompt and program stands for a series of pattern and actions.

When the program is long, it is usually more convenient to put it in a file and run it with a command like this:

```
$awk -f program-file input-file1 input-file2
```

The *awk* utility reads the input files one line at a time. For each line, *awk* tries the patterns of each of the rules. If several patterns match then several actions are run, in the order in which they appear in the *awk* program.

Processing HTML Files with *Gawk* Scripts

Three documents are currently presented in HTML format in the ERL. They are: *Construction Manual*, *Standard Specifications*, and *Supplemental Specifications*. As they are compiled in different word processing programs (MS Word™ for *Construction Manual*, and Corel WordPerfect™ for the others), different *gawk* scripts are needed to process the respective HTML files. Here we will use the *Standard Specifications* case to illustrate how to develop a gawk script to process the HTML files.

The input HTML files to be used in this example were converted from WordPerfect files using WordPerfect 8.0. Five types of changes will be made to these HTML files:

1. Divide the input files, which contain one division each file, into smaller files, which contain one section each. Division files are named Div_##.htm, say Div_25.htm. Section files are called ####.htm. Here, # refers to a digit between 0 and 9.
2. Generate the <head>…</head> part, including title, etc, for each of the section files.
3. Delete unnecessary format tags.
4. Add bookmarks to section names, subsection names, and subtitles.
5. Add links to the references.

In this thesis, we used the gawk program that comes along with Cygwin, a UNIX emulator for Windows systems. To perform these modifications to the input files, a UNIX shell script called "gshtm.sh" (see **Appendix A**) was developed. The shell script primarily consists of four parts of gawk scripts. Based on the functions these gawk scripts perform, they can be referred as "pre-format", "head", "body", and "end".

A. Pre-format

In most cases, the immediate output HTML files converted from WordPerfect format contain some unwanted or redundant format tags (Figure 4-2). Because gawk scripts depend on patterns to decide what actions should be taken to the file contents, these unwanted tags should be cleared and the codes in the HTML files should be simple and correct. Thus a pre-format process is necessary to make the input files more controllable and the patterns easier to identify.

A short script is written to identify and delete these tags (See **Pre-format** part of **Appendix A**). After the pre-format process, the HTML codes looks as shown in Figure 4-3.

The output files of this process are named Div_##.html and will be used in later processes.

```
<HTML>
<HEAD>
<META NAME="Generator" CONTENT="Corel WordPerfect 8">
<TITLE></TITLE>
</HEAD>
<BODY  TEXT="#000000"  LINK="#0000ff"  VLINK="#551a8b"  ALINK="#ff0000"
BGCOLOR="#c0c0c0">

<P><CENTER><STRONG>DIVISION          20.                    EQUIPMENT
REQUIREMENTS</STRONG></CENTER>
</P>

<BR WP="BR1"><BR WP="BR2">
<P><FONT SIZE="-1">This Division consists of requirements for equipment
used on various types of
construction  and  maintenance  as  set  forth  in  the  following
sections.</FONT></P>
```

Figure 4-2 HTML File Codes Converted from WordPerfect Files by WordPerfect

```
<HTML>
<HEAD>
<META NAME="Generator" CONTENT="Corel WordPerfect 8">
<TITLE></TITLE>
</HEAD>
<BODY>
<P><CENTER><STRONG>DIVISION          20.                    EQUIPMENT
REQUIREMENTS</STRONG></CENTER>
</P>
<P>This Division consists of requirements for equipment used on various
types of
construction and maintenance as set forth in the following sections.</P>
```

Figure 4-3 Content of the HTML files after being processed by "Pre-format"

B. Head

This part reads the preformatted HTML files as input, identifies the section name (####), generates a file name using the section name, and creates the Head part of HTML files (see **HEAD** part of **Appendix A**).

Two types of section HTML files will be generated for the *Standard Specifications* document. The first type is named as ##00.htm, which contains the introduction part of each division file. To generate the filename for this type of files, the script identifies the division

name, i.e. 20, from the input script file; add "00" to the division name; then a file called 2000.htm can be generated.

The second type contains the main contents of the sections. The section names are identified from the section titles.

The lines containing division and section titles are written in the following way:

```
<P><CENTER><STRONG>DIVISION 20.  EQUIPMENT REQUIREMENTS</STRONG></CENTER>
<P><STRONG>2001.01  GENERAL.</STRONG></P>
```

Figure 4-4 Division/Section Title Patterns

In the input files, a unique pattern exists for lines containing division and section titles.

For division titles, they contain tags "<P>", "<CENTER>", "<STRONG>" and texts "DIVISION ##.". For section titles, they contain tags "<P>", "<STRONG>" and texts "####.##" immediately after the "<STRONG>" tag. Thus whenever the script comes across these patterns, the computer knows a new division or section begins. By using string manipulation functions provided by gawk, the critical numbers and strings that will be used to generate the filenames and the title tag for the section files can be trimmed out. Then a few print commands will generate the head part for each section file using the correct file name.

C. Body

This part generates the <BODY> part of the section files. Two types of modifications are made to the HTML codes:

- Add bookmarks to subsections and subtitles
- Add hyperlinks to texts referring to other sections

The logic behind the scripts can be described by a flowchart as shown in Figure 4-5.

Bookmarks are to be added to two types of titles in the HTML files: subsection titles and subtitles. A subsection title goes like this:

```
<P><STRONG>2501.08  TEST PILES.</STRONG></P>
```

A subtitle goes as follows:

```
<P><STRONG>A.      Gravity Hammers.</STRONG></P>
```

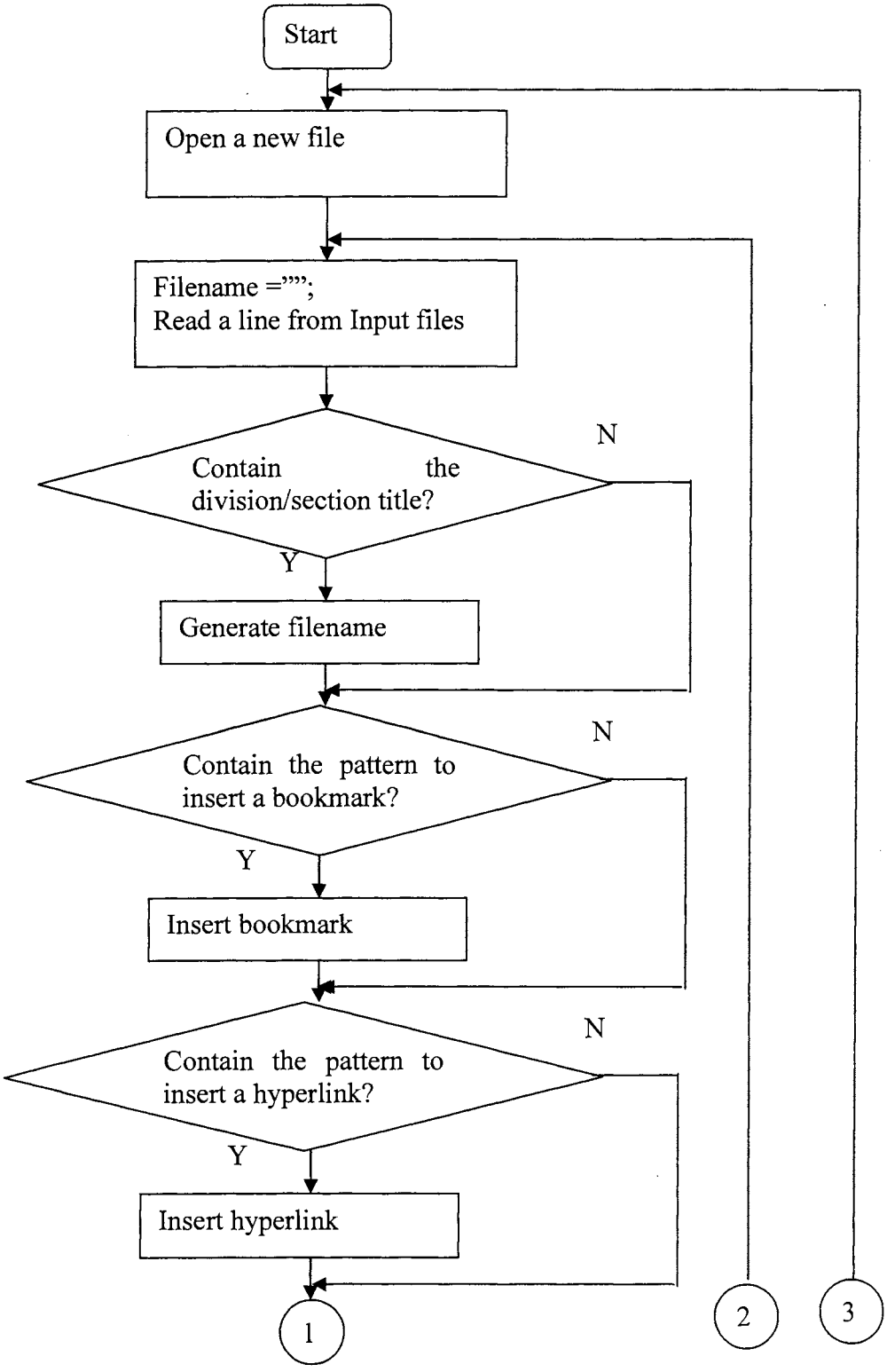


Figure 4-5 Flowchart for scripts creating "Body" part of section files

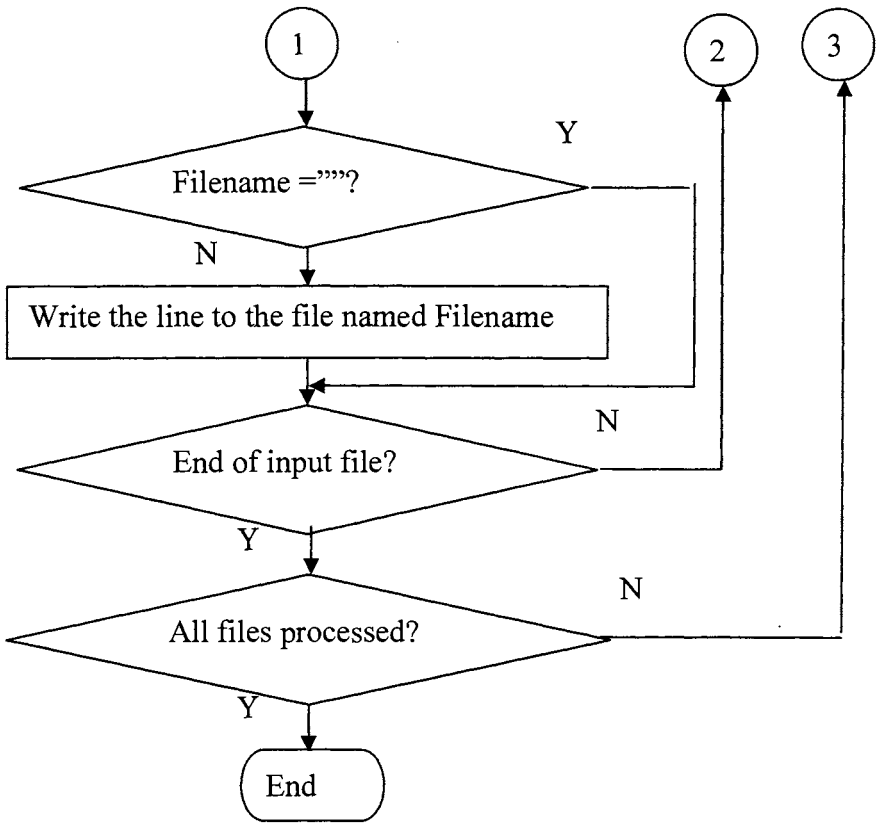


Figure 4- 5 (Continued)

As no other texts use the same format, thus patterns that uniquely define such titles can be extracted from such examples.

The mechanism to insert hyperlinks is a little tricky and deserves more explanation. A few variations exist in the texts referring to other specification code. Types of patterns to be recognized are listed below:

1. `Section 2547;`
2. `Article 1107.09;`
3. `Articles 1107.08, 1107.09, and 1108.03;`
4. `Materials I.M. 213 and 214;`
5. `Articles 1107.08, D, E, and F;`
6. `Articles 1107.08, Paragraph E.`

The difficulty is with situation 3 and 5, and combinations of the two. To correctly specify such types of patterns, a mechanism is devised. As an example, the flowchart of

Situation 5 can be treated similarly. The script corresponding to this flowchart is shown in Figure 4-7.

D. End

This part of the gawk scripts writes ending codes, such as "</BODY>" and "</HTML>" to each section file.

When running the shell script, HTML files created by Corel WordPerfect 8.0 and the shell script should be placed in the same directory. After the shell script is run in Cygwin, a subdirectory called "content" will be created containing all sectional files with bookmarks and hyperlinks inserted. The subdirectory is ready to be copied to the working directory where the new version of ERL is compiled (Please refer to Section 5.1 Standardizing ERL File Storage Structure for the ERL file storage structure).

To make the scripting process simpler, a simple and standardized file storage structure and a strict naming convention would be of great help. For example, instead of storing the section files by division, storing all the content files in one directory would make the hyperlinks that refer to another section file much simpler both in the *gawk* scripts and the HTML codes.

Another way to make the gawk scripting simpler is to introduce HTML 4 in the HTML document content files. HTML 4 makes the HTML file much more structured. For example, instead of identifying the division title by its presentation tags(, etc.), we can edit the Word files in a certain way so that a unique tag or a "class" name can be assigned to the line containing the division title, i.e. <H1> / <P class=DivisionTitle>, which will make the pattern very easy to identify.

Improvements to the ERL in these two aspects will be discussed in Chapter 5.

Besides adding bookmarks and links in the converted HTML files, *awk* scripts can also be used for other purposes, such as creating the Quick Jump JavaScript codes (which is now stored in the file called "nav.js" under the "navigation" directory).

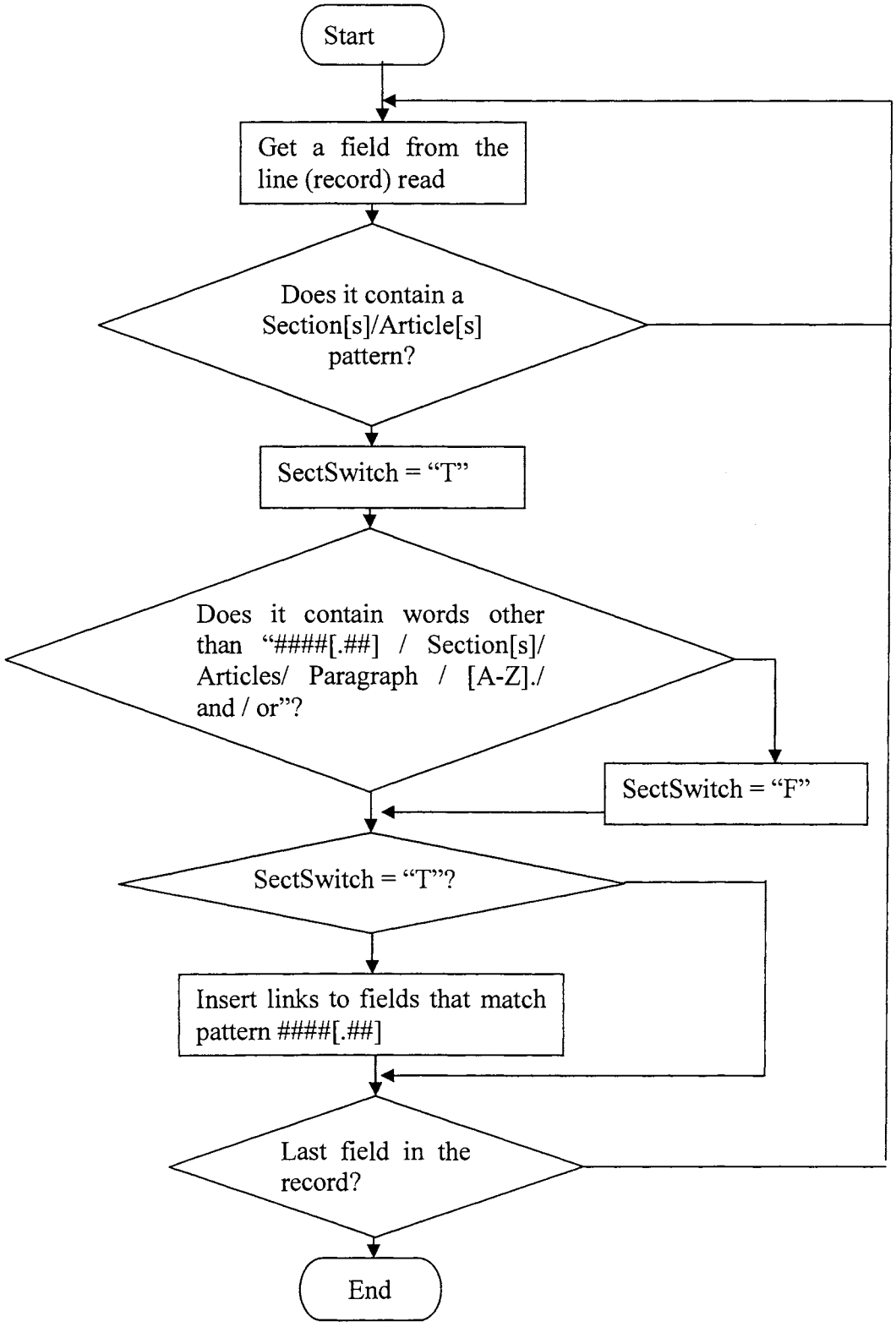


Figure 4-6 Insert Links for Situation 3

```
for (i=1;i<= NF; i++)     {
      if (((($i ~ /Article[s]*/) || ($i~/Section[s]*/) ) && ($0 !~
/<CENTER>/ )) { SectSwitch = "T"}
      if
((($i!~/Article[s]*/)&&($i!~/Section[s]*/)&&($i!~/Paragraph/)&&($i!~/[
A-Z]([^A-Za-z]|$)/)&&($i!~/and/)&&($i!~/or/)&&($i!~/[0-9][0-9][0-
9][0-9]/))
            {SectSwitch="F"}
        if ((SectSwitch=="T") && ($i~/[0-9][0-9][0-9][0-9]/))      {
            if ($i~/[0-9][0-9][0-9][0-9]\.[0-9][0-9]/)             {
                  subSectSwitch="T";
                  mat=match($i,/[0-9][0-9][0-9][0-9]\.[0-9][0-9]/);
                  flname=substr($i,mat,4);
                  ssname=substr($i,mat,7);
                  sub(ssname,                                      "<a
href=\""flname"\.htm#"ssname"\">"ssname"<\/a>", $i);
                                                          }
            else              {
                  if ($i~/[0-9][0-9][0-9][0-9]/)             {
                        mat=match($i,/[0-9][0-9][0-9][0-9]/);
                        flname=substr($i,mat,4);
                        sub(flname,                                "<a
href=\""flname"\.htm\">"flname"<\/a>", $i);
                                                   }
                                  }
                                                   }
}
```

Figure 4-7 Gawk Script Codes That Insert Links Where Situation 3 Happens

Using *Perl* for PDF files

In the current ERL, the April 2002 version, two and a half of contract documents (*Standard Road Plans*, *Instructional Memorandums*, and Appendixes of *Construction Manual*) are presented in PDF format. Before *Perl* was employed as a tool to the ERL project, checking hyperlink errors in the PDF files and restructuring the ERL is the last thing an ERL maintainer would want to do, because he/she would have to run through each of the PDF files (currently about 1000 of them for each letting date) and check every link in it to see whether the link is influenced by the restructuring and if the link works properly.

After reviewing the PDF file characteristics and capabilities of *Perl*, a *Perl* script was written to generate a list of all hyperlinks in the PDF files. By checking the way the links are written, we can easily identify whether an error exist in the links. Another use of *Perl* is in restructuring the ERL. Suppose we need to change the directory name of "conmanual" to

"cm". We can write a *Perl* script which goes through each line of the PDF files, finds the pattern "conmanual", and changes it to "cm".

4.2 Documenting the Development and Maintenance Procedure

One of the issues that come along with the long time span of the ERL life cycle is the turnover of project team members. Currently, key ERL updaters are graduate and undergraduate students form Iowa State University and IDOT employees. Majority of them will leave the project after working on the project for about two years. Though most skills needed in ERL maintenance and update are not very complicated, they are not typical skills for these people. Each time a new member joins the team, it normally takes about half a year for these new members to master the skills and get acquainted with the update procedure, and they are very likely to make mistakes on their first update task. To reduce the time of learning and cost of errors, a document called *Instruction Manual for ERL Development and Maintenance* (*Instruction Manual*) is compiled to assist successive updaters.

Before the *Instruction Manual* was compiled, the only documentations that recorded ERL development and maintenance procedures were the theses written by earlier graduate students. While these theses can provide helpful information for later project members, they cannot take the place of the *Instruction Manual*. Unlike a thesis, which normally focuses on the particular tasks of a graduate student, the *Instruction Manual* should document all activities involved in the development and maintenance process completely and in sufficient detail to allow succeeding students to learn quickly. Moreover, the *Instruction Manual* is more interested in how a certain task is accomplished without paying much attention to why this solution is selected.

Once a solution is written in the *Instruction Manual*, it becomes a standard solution to be followed by later maintainers. Thus, the solution must be carefully tested and must prove effective and efficient before being included in the ERL.

Topics discussed in the Instruction Manual include:

1. Overview of ERL maintenance process
2. Preparing for ERL update
3. Creating document files

4. Working with navigation functions
5. Working with the Cover Page
6. Error checking

The compilation process was an interactive effort between the compilers and intended readers. A draft of the *Instruction Manual* was first prepared by ISU researchers. The draft was used for learning the update methods by two project members at IDOT who recently undertook the updating task. Problems and suggestions they found when following the instructions of the manual were collected and revisions were made to the *Instruction Manual.*

The learning experience of new maintainers from IDOT proved the effectiveness of using *Instruction Manual* as a means of training new ERL maintainers. The learning time is significantly shortened and consistency of ERL functions and appearance is better protected.

The *Instruction Manual* records the best practice of ERL development and maintenance available to the compilers. As time goes by, new tools and methods may be found by successive updaters. Thus revision to the Instruction Manual will be an ongoing effort as long as new versions of ERL are to be published.

4.3 Using Project Specific Web Site (PSWS) for ERL Development

Introduction

WWW is gradually accepted as a helpful tool for construction project management because of its unique capability of distributing and collecting information. Though the ERL project is not as complicated as a construction project, information communication has been a troublesome issue because:

a. Team members work at different organizations, though in the same town.
b. More than one office at IDOT is involved in the ERL project.
c. The turnover rate of the project team is high because most updaters are graduate and undergraduate students from ISU. They will leave the project when they graduate. Communication of knowledge and skills from early researchers to later team members is a problem in the ERL project.

d. Activities of IDOT offices and ISU researchers are interactive to each other. The task of one participant may constitute the prerequisite of another. Coordination among team members is important to project performance.

A three step approach of developing a web-based project information system is suggested in an essay prepared by engineers at 3D International, a firm providing Architecture/Engineering and Program Management services:

1. Understanding the fundamentals of Internet technology.
2. Determine specific project needs.
3. Make it happen.

Considerable effort has been spent on understanding the fundamentals of Internet technology in this thesis. We will discuss the information need for the ERL project specific web site (PSWS) and the actually implementation methods adopted.

Project Information Needs

After considering the challenges as described above and the capabilities of PSWS, the following features are selected as requirements for the ERL PSWS:

- The PSWS should be able to serve as a knowledge base.
- The PSWS should be able to provide progress control functions.
- Provide a centralized daily log system for ERL update activities.
- Provide a centralized error report system.

Beyond these features, the web site should also have the following attributes to make the ERL PSWS practically helpful for the project:

- The PSWS should be easily updated by authorized team members.
- Information stored on the web site should be well structured and easy to retrieve.
- Little maintenance effort should be needed.

Implementation Method

In this stage, it is necessary to make a few specific decisions on how the ERL project website should be implemented. These decisions are discussed below:

A. Hardware and Software: As the number of target users of the project website (project team members) is quite small, the chance of these members using the website at the same time is very low. Thus, a personal computer equipped with a 200MHz Pentium pro CPU and 96 MB RAM, which was available to the research team, was used to server as the web server. After having experienced a few hacker attacks in another Web Server which uses Microsoft IIS, researchers decided to use Redhat Linux/Apache, which has a better security record, as the Operation System and web server.

B. Content Architecture: The website is divided into four sections:
   - Knowledge Base: This section contains documents written by project members about ERL development and management, and links to Internet resources which contain helpful information on generic web authoring and programming, text conversion, photo editing, etc.
   - Products: This section contains all versions of ERL that have been published up to date and ERL's published by other DOT's.
   - Related Links: This section contains links to websites where technical information can be obtained, and links to DOT's which maintains online versions of the ERL.
   - Contact Information: This section lists the names, phone numbers, email addresses, and working addresses of project members.
   - Discussion Board: A discussion board system is used in the website for its interactive capability. Schedules, work progress, daily logs, and suggestions for new improvements can be posted on the discussion board in a real-time manner and with as little effort as possible from the team members.

C. Authoring and scripting tools. Contents in the Knowledge Base, Products, Related Links, and Contact Information section don't have to be updated frequently. They will be written in HTML and CSS. Microsoft FrontPage 2002 is used to edit these web pages. By using the "theme" function provided by FrontPage 2002, a consistent style can be generated with ease.

D. Implementing the Discussion Board. The Discussion Board can be implemented using CGI scripts written in Perl or other languages. Because many ready made CGI resources can be freely available from the Internet, a set of CGI script codes for discussion board, called "YABB", is selected for the ERL project web set. The discussion board is customized to include a few categories and boards to make information posted on it more structured and easier to be retrieved. The boards that are set up for people to post include:
    a. Useful resources: for people to post links to resources they found helpful;
    b. Schedules: Schedules, deadlines, etc settled on meetings;
    c. Daily Logs: for project members to report their daily progress;
    d. Error Report: for project members to report errors they found in the ERL;
    e. Suggestions: for project members to post opinions about future development in the ERL.

As a result, a project web site is created for ERL development and maintenance at http://erl.cce.iastate.edu/ERL/. Similar to the diffusion paths of many other new technologies and tools, recognition of the project web site takes time. A few team members from ISU have used the PSWS in the April 2002 ERL update in reporting working progress, and errors found. Some positive effects have been observed. Further effects of using PSWS in improving the ERL project performance remains to be discovered through future updating process.

4.4 Collecting Users' Feedback

Information from the intended users of the ERL is a significant factor to be considered in ERL development and maintenance. Such information includes:

1. What documents published by IDOT are most frequently used?
2. What are the most desired attributes of the ERL?
3. What kinds of hardware do the users use in terms of media storage, hard drive space, CPU speed, types of Network Interface Card or Modem, etc?
4. What kinds of software do the users use in terms of Operating System, type and version of browser, etc?

5. What is the preferable distribution method?
6. How do they use the ERL? For example, from the CD-ROM, copy the ERL to hard drive, or via the Internet?
7. What is the most commonly used way of message retrieval? Through the navigation bar, the search engine, or the Quick Jump function?
8. Errors and bugs found when using the ERL.
9. Suggestions and new ideas for ERL improvement.

A few interviews, focus group meetings and online surveys had been conducted to gather users' information in earlier research stages. They are not enough. As many factors concerning the users change through the years, what users want from the ERL will also change. The same users will definitely give different answers to at least some questions listed above after a few years. Thus a continuous effort of collecting users' response is necessary for effective ERL maintenance. Many methods are available for collecting information from users and clients. Besides the ones used in earlier stages and discussed in other theses, a few others are also identified as potentially helpful for ERL maintenance. These methods include:

1. Mail survey;
2. Usability testing;
3. Web-server stats;
4. E-mail.

While all these methods are for collecting user input, they are intended for different types of information and subgroups of users. Introduction of each method and why they are suggested for the ERL project is discussed below.

Survey by Mail

Survey is a good method for collecting information from intended users. Survey can be conducted through a number of ways, such as in person, telephone, through the Internet, and by mail. Mail is suggested here because of the following reasons:

A. Equal accessibility to all intended users

Different level of accessibility to the Internet and phone service may exist among ERL users. Some of the users may have broadband and all time access to the Internet through a LAN or DSL, while others my still use 28.8K dial-up modem. Thus it is very likely that those who use 28.8 K dial-up modem are less interested in responding to an online survey, which will make the information collected from the survey not come from a valid sample of the target population. This is not an issue for mail, or at least not an issue significant enough to cast doubt on the survey result.

B. Good Sample Control

Each time a new version of ERL is released, about 1000 copies of the ERL CD's are sent out to users by mail. A user database which contains basic information (company name, mail address, etc) about the users is maintained at IDOT. By enclosing a questionnaire and a stamped envelope with the CD to selected users, we can easily control the sample of the survey.

C. Reasonable Cost

Some cost is expected to be spent on paper, envelopes and stamps. Depending on the planned sample size needed, the expense ranks about a few hundred dollars, which is an acceptable price for the ERL project.

D. Reasonable Response Time

As most target users dwell and work within Iowa, the response time including the mailing time is estimated to be about two weeks, which is acceptable considering the maintenance cycle of the ERL.

E. Acceptable Difficulty in Data Processing

Additional effort might be needed to input data into the computer compared with an online survey. As not many information are collected and the estimated response ranks between a few dozens and one hundred, it may not be too much a burden for the ERL team members.

Though many disadvantages exist for the mail-based survey compared with online survey, such as higher cost, longer response time, and more work in data processing, these disadvantages are minor and not unacceptable to the ERL project. The mail-based survey is suggested here primarily for it offers equal opportunity for users to respond and thus assures correctness in the survey result. Unless survey result proves that all users have equal accessibility to the Internet, mail-based surveys are preferable to online surveys.

Usability Testing

Before introducing the procedure of usability testing, an understanding of usability and usability testing would be helpful for us.

A few definitions of usability are available, anyone of which is enough to give us a concept of what the term means. One definition given by International Organization for Standardization in ISO 9241-11 goes like this:

> **The extent to which a product can be used by specified users to achieve specified goals in a specified context of use with effectiveness, efficiency, and satisfaction.**

Jacob Nielsen identifies the following components, or attributes of usability (Usability Engineering, 1993):

- Learnability. The system should be easy to learn so that the user can rapidly start doing some work.
- Efficiency. The system should be efficient to uses, so that once it is learned, the user can achieve a high level of productivity.
- Memorability. The system should be easy to remember, so that the casual user is able to return to the system after a time and not have to learn it over again.
- Errors. The system should have a low error rate, so that users make few errors and can easily recover from them.
- Satisfaction. The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it.

The process of learning from users about a product's usability by observing them using the product is called *usability testing* (Barnum, 2002). Usability testing generally has the following characteristics (Dumas and Redish, 1993):

1. The primary goal is to improve the usability of a product. For each test, there must be specific goals and concerns that you articulate when planning the test.
2. The participants represent real users.
3. The participants do real tasks.
4. The team observes and records what participants do and say.
5. The team analyzes the data, diagnoses the problem, and recommends changes to fix the problem.

Traditional approach of usability testing, as it was commonly practiced well into 1980's, was expensive and time-consuming because tests were conducted in labs managed by usability experts and large number of "subjects" were needed (Barnum, 2002). Later research and experience shows that "the maximum benefit-cost ratio is achieved when using between three and five subjects" (reported in Nielsen, "Guerrilla HCI" 251). It was also determined that the test did not need to be recorded in a lab, because note takers could capture the essential findings by hand (Barnum, 2002). Even if recording equipments are needed, portable video cameras with long duration batteries can be set up wherever the test is to be practiced.

Usability testing is appropriate for ERL maintenance because usability is one of the primary concerns for ERL development. Resources and conditions for conducting usability testing can be easily achieved at reasonable cost and in a timely manner. "Subjects", users to be tested, can be found from IDOT designers and inspectors.

Web-server Statistics

All web server packages allow web site administrators to track visits to the website. There are two types of logs that every web server should use (Stout, 1997):

- Transfer (or access) log
- Error log

Some web servers allow web administrators to achieve additional data through the following types of log files:

- Referrer log
- Agent log

The actual pathname and filename used for the log files may be different system from system, but they generally provide the same types of data for analysis. As an example, we will discuss how to log visits on Web Site powered by Microsoft Internet Information Server (IIS) 5.0 in a Microsoft Windows 2000 Professional environment.

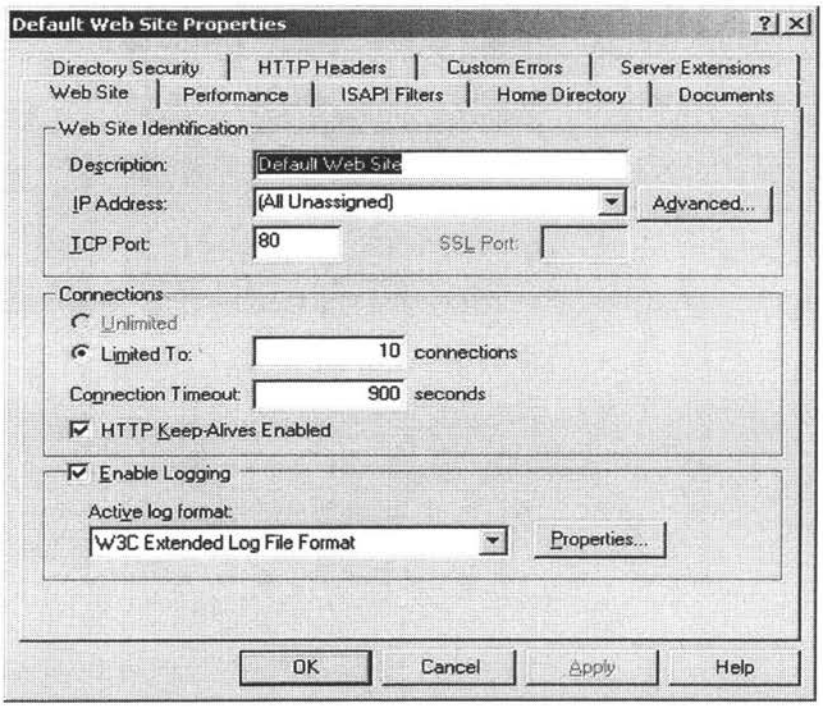


Figure 4-8 Enabling Logging Function for IIS 5.0

After IIS 5.0 is properly installed, we can enable the logging function for IIS by following the steps (adjusted from:http://www.microsoft.com/windows2000/en/server/iis/) listed below:

1. Open **Internet Service Manager** by selecting **Start – Control Panel – Administrative Tools – Internet Service Manager**.
2. Select a Web or FTP site, and open its property sheets.
3. On the **Web Site** or **FTP Site** property sheet, select the **Enable Logging** check box.
4. In the **Active log format** list, select a format. By default, the **Enable Logging** check box is selected and the format is **W3C Extended Log File Format** (See

Figure 4-8 Enabling Logging Function for IIS 5.0), with the following fields enabled: **Time**, **Client IP Address**, **Method**, **URI System**, and **HTTP Status**.

5. **Note** If the format you select is ODBC logging, click **Properties** and then type the Data Source Name and the name of the table within the database in the boxes. If a user name and password are required to access the database, type these also and click **OK**.
6. Click **Apply**.
7. Click **OK**.

As the **W3C Extended Log File Format** is a commonly accepted format and serves the needs for the ERL project, we will use it in this example. Information available from website log file analysis is shown in Table 4-1.

Table 4-1 Information Available from IIS Log File Analysis (http://www.openwebscope.com/)

Daily Activity	Most Accessed Web pages
Last Visitors	Least Accessed Web pages
Hostnames	Entry Pages
Domains	Exit Pages
Countries	Paths Through Site
File Requests	Web Browsers
File Types	Operating Systems
Directories	Search Engine Keywords
Hourly Summary	Referrer Pages
Monthly Summary	Server Status Codes
Authenticated Users	Broken Links
Search Engine Spiders	

Depending what the webmaster wants to know, he/she can customize the types of data to be collected in the log file. To do this (http://www.microsoft.com/):

1. From **Internet Service Manager**, Select a Web or FTP site and open its property sheets.

2. Enable logging if it is disabled and select the "W3C Extended log file format".
3. Click **Properties**.
4. On the **Extended Properties** property sheet, select the fields you want to log. By default, **Time**, **Client IP Address**, **Method**, **URI Stem**, and **HTTP Status** are enabled.
5. Click **Apply**.

A log file is a text file that records the activities performed on a web server when a user visits a web site (Figure 4-9). To obtain information from these text files, statistical analysis is required. Fortunately, some software packages are available for website log file analysis. One of such packages called *OpenWebScope*™ is found to be useful and will be introduced briefly as an example of how to analyze web site log files (http://www.openwebscope.com/).

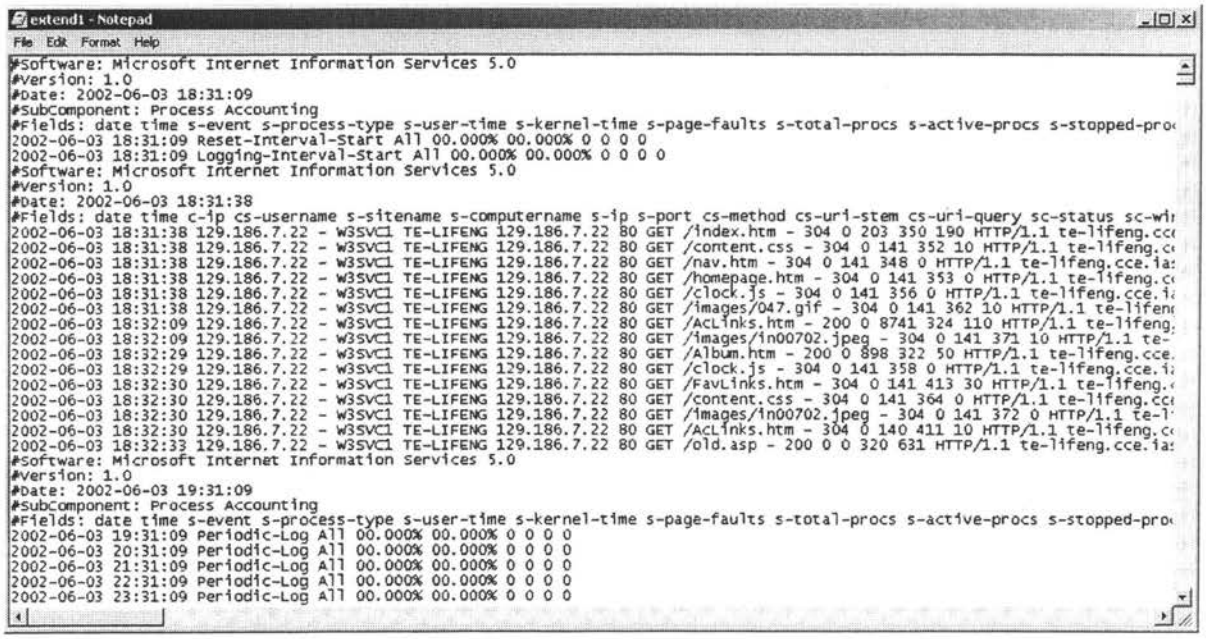


Figure 4-9 Log File Generated by IIS 5.0

After having installed the program on the computer, we can start the program by selecting "**Start – Program Files- OpenWebScope – OpenWebScope Web Statics**". Then the following window will appear. By selecting New either from the **Site** menu or by pressing the button and inputting a few customization information, such as name of web sit and path of log file name etc, we can generate a report for the web site we want analyze.

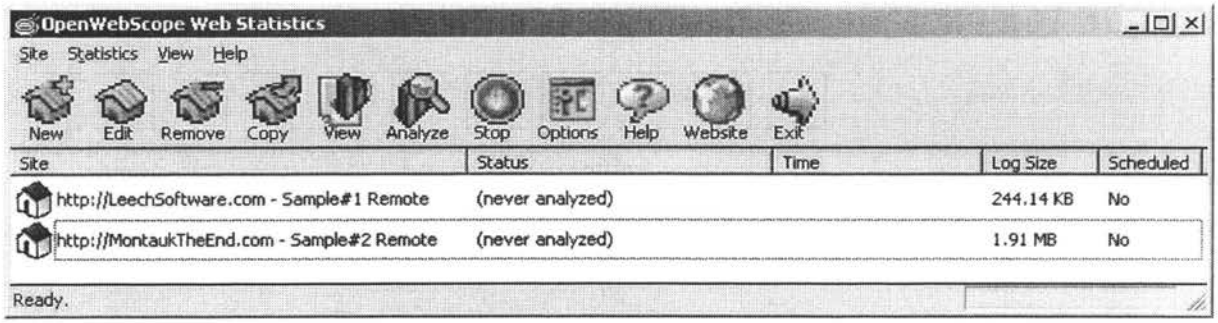


Figure 4-10 OpenWebScope Web Statistics Interface

An example of how the web statistics report will look like is shown in Figure 4-11.

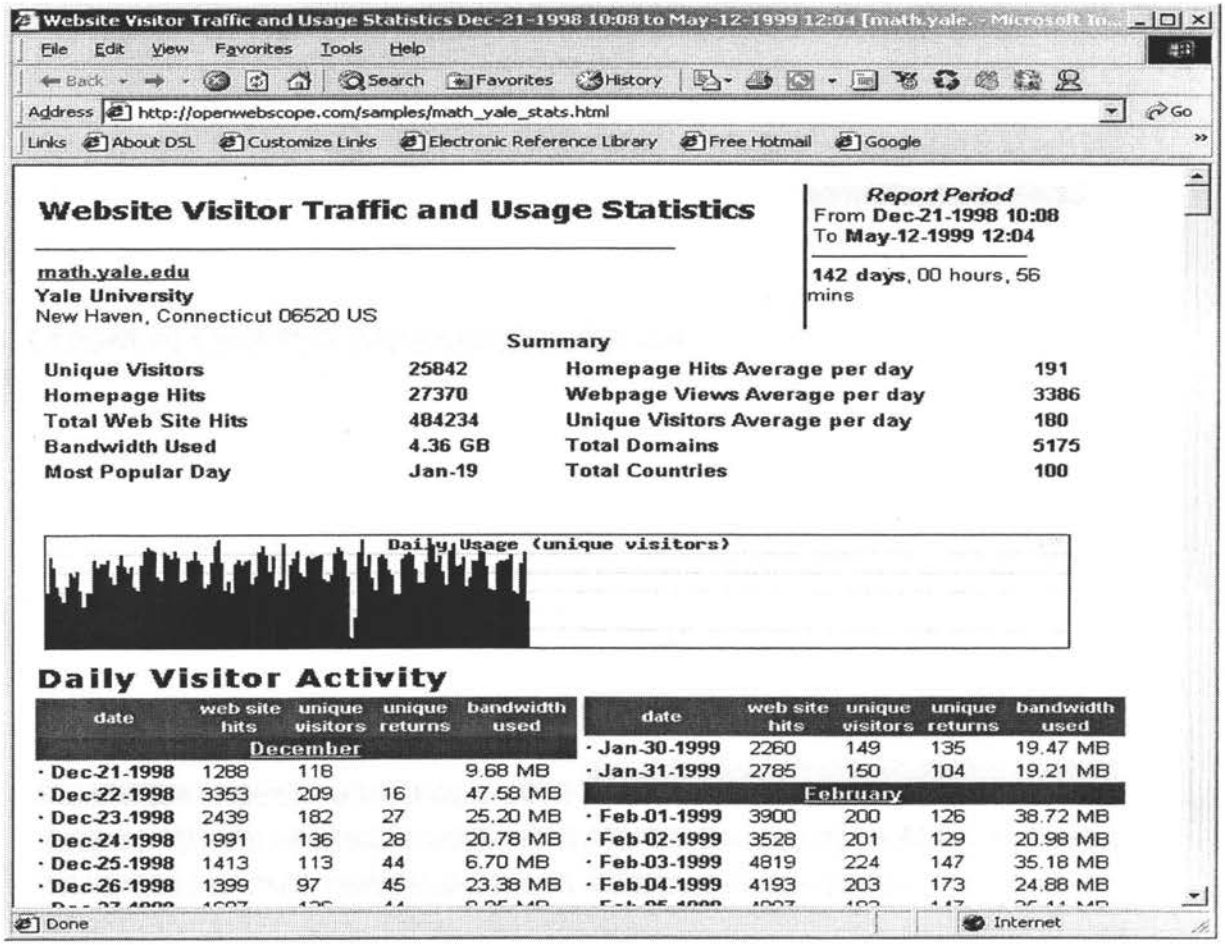


Figure 4-11 Sample Web Statistics Report Generated by OpenWebScope

The method of collecting user feedback from website log files is suggested for the ERL project because many types of information collected this way are useful for ERL maintenance and development. Such information types include:

1. Information about the browser and Operating System used;
2. What types of documents are most commonly used?
3. What types of documents are least used?
4. Errors and bugs;
5. Users' means of Internet access;
6. Navigation methods used by users;
7. Is the Internet a popular method for ERL distribution?

The primary condition that this method can be used in the ERL project is that the ERL could be available online. Fortunately, it is. The latest version of ERL can be accessed at IDOT's website. Thus the works of conducting an analysis will be to configure the Web Server and analyze the data.

E-mail

The convenience, cost efficiency and timeliness of e-mail have attracted so many people that almost everyone in the United States has at least one e-mail account. Many organizations use it for customer support and collect users' feedback. While the above described methods are good at collecting predefined information through a relatively fixed period of time, some useful information may not be easily qualified, quantified, categorized or known when it is available. As a complementary to the above methods, a devoted e-mail account can be used for collecting such type of user responses.

CHAPTER 5. IMPROVING MAINTAINABILITY OF THE ERL

By maintainability, we refer to the internal quality of the ERL that maintenance, such as modification and update, can be made to the ERL with minimum amount of work and as little chance of error as possible. The issue of improving maintainability needs to be considered in conjunction with the maintenance methods and should not sacrifice user desired attributes as an expense. This chapter discusses issues about improving maintainability of the ERL in two aspects:

- Improving the file storage structure, and
- Making the HTML file more structured by using HTML 4.

5.1 Standardizing ERL File Storage Structure

Creating the file storage structure is one of the major works in ERL implementation. A well designed file structure is of special importance in ERL development because:

1. The file structure must be determined at an early stage in ERL development, so that hyperlinks can be made in the document files;
2. Once determined, the file names and directory names cannot be changed arbitrarily, as any change in the file names and directories may invalid many links in the document files. In most cases, changes in the file structure will cause great amount of work in changing links in the HTML and PDF document files and will very likely lead to errors.
3. ERL maintainers need to understand the file structure to update the ERL. Therefore the file structure should be simple, logically reasonable and easy to remember.
4. The file structure should resemble the navigation structure.
5. The file structure should serve the objectives of the ERL. A well designed file structure will be contributive to many desired attributes of the ERL such as flexibility and portability, that is, the ERL or any component of it can be easily used for other purposes and in Operating Systems other than Windows.

The file structure used in earlier versions of the ERL is shown in Figure 5-1.

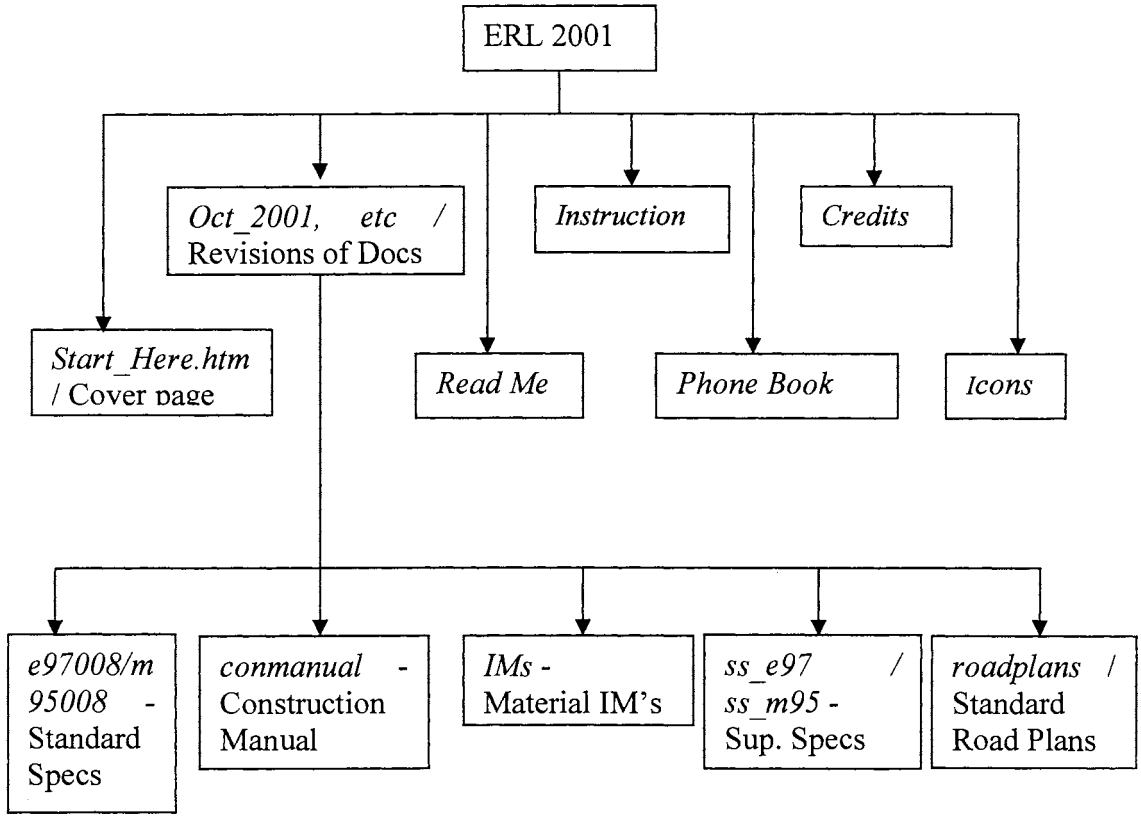


Figure 5-1 File Structure Used in Earlier ERLs

Although the file structure used in earlier versions of the ERL works satisfactorily, a few defects were found and further improvements were deemed to be desirable. Defects in the original file structure include:

1. While the file structure for each document in the ERL shares many common features and a standard file structure can be developed for all documents, such a standard structure was not developed (compare Figure 5-2 and Figure 5-3 for example). Files and folders are named arbitrarily. This will very likely lead to errors or inconveniences when updaters make links in the document files. It also makes understanding the file structure more difficult for new updaters. Instead of learning one standard file structure for all documents, the updater must try to study five file structures one by one.

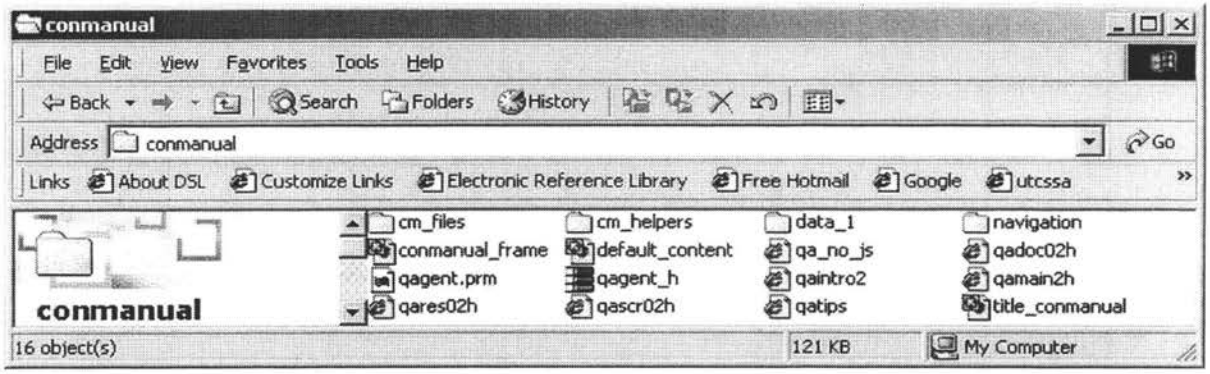


Figure 5-2 File Structure for Construction Manual (ERL October 2001)

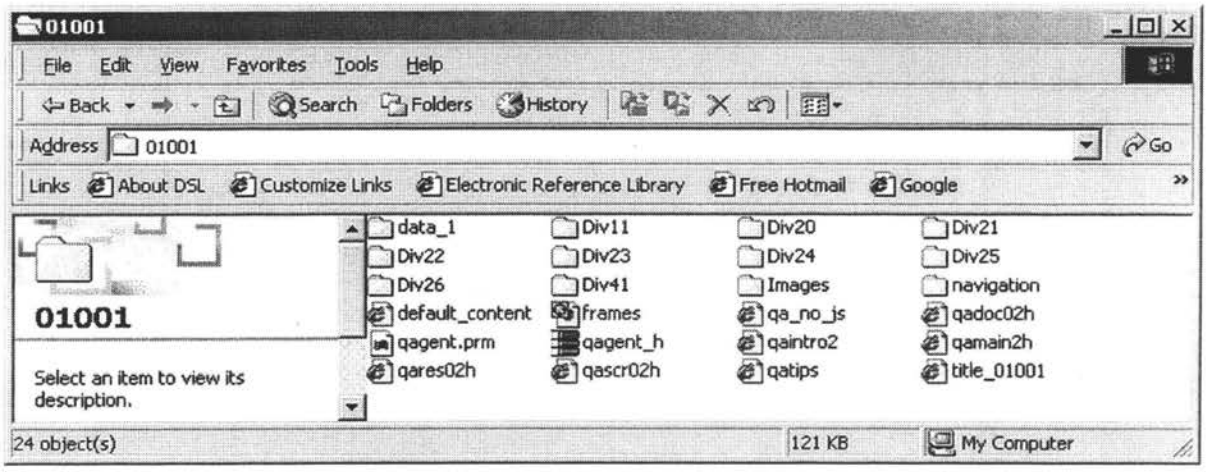


Figure 5-3 File Structure for Standard Specifications (ERL October 2001)

2. Portability is not fully supported by the current ERL file structure. By portability, it is meant that the ERL should work properly in many types of operating systems. Currently the problem with the ERL files structure is that it does not function properly when used on a UNIX web server. The ERL was originally developed in the Windows environment which neglects the letter cases. For instance, by referring to "frames.htm", we can link to a file called "Frames.htm" in the Windows system. But if the CD is hosted on a Unix/Linux web server, this link will be broken because Unix/Linux commands are case sensitive.
3. The complexity of the original file structure makes it inconvenient to process the HTML files with *gawk* and *Perl* scripts.

Based on the ideas listed above, a standard file structure and a naming convention for ERL document files are proposed. The standard file structure is shown in Figure 5-4.

The naming conventions are listed below:

1. Lower case. All file and directory names are suggested to use lower case. Hyperlinks in files should use lower case too. This is to avoid chances of errors when the ERL CD is put on a web server.
2. Use descriptive two letter abbreviations for the directory name for each document. Instead of using "conmanual", "roadplans", etc, a two letter naming system for the documents collected in an ERL will be large enough to identify the document names and simple enough to avoid mistakes. To date, the names for the documents in the ERL are:
   a. "gs" for *Standard Specifications with Revisions*;
   b. "ss" for *Supplemental Specifications*;
   c. "rs" for *Standard Road Plans*;
   d. "cm" for *Construction Manual.*
   e. "im" for *Material Instructional Memorandums*;
3. Document Opening Pages. The Document Opening Pages refer to the HTML pages defining the frameset and the appearance of each frame. For all the documents included in the ERL, these files are named in the following way:
   a. "frames.htm" is the file that defines the layout of the frameset. The frameset consists of three parts, the "title" bar, the "navigation" bar, and the main frame.
   b. "title.htm" is the file to be shown in the "title" bar.
   c. "default_content.htm" is the file that is to be shown in the main frame.
   d. Content in the navigation bar are navigation files, the naming convention of which is discussed in item 4.
4. Navigation files are stored in a directory called "navigation". Use simple and consistent names for the navigation files, such as "nav11.js".
5. Document files are files that contain the main contents of construction documents. These files are saved directly in a subdirectory called "content". To avoid

complexity in hyperlinks, use logical and simple names for the document files, such as "760.htm" (for CM 7.60).

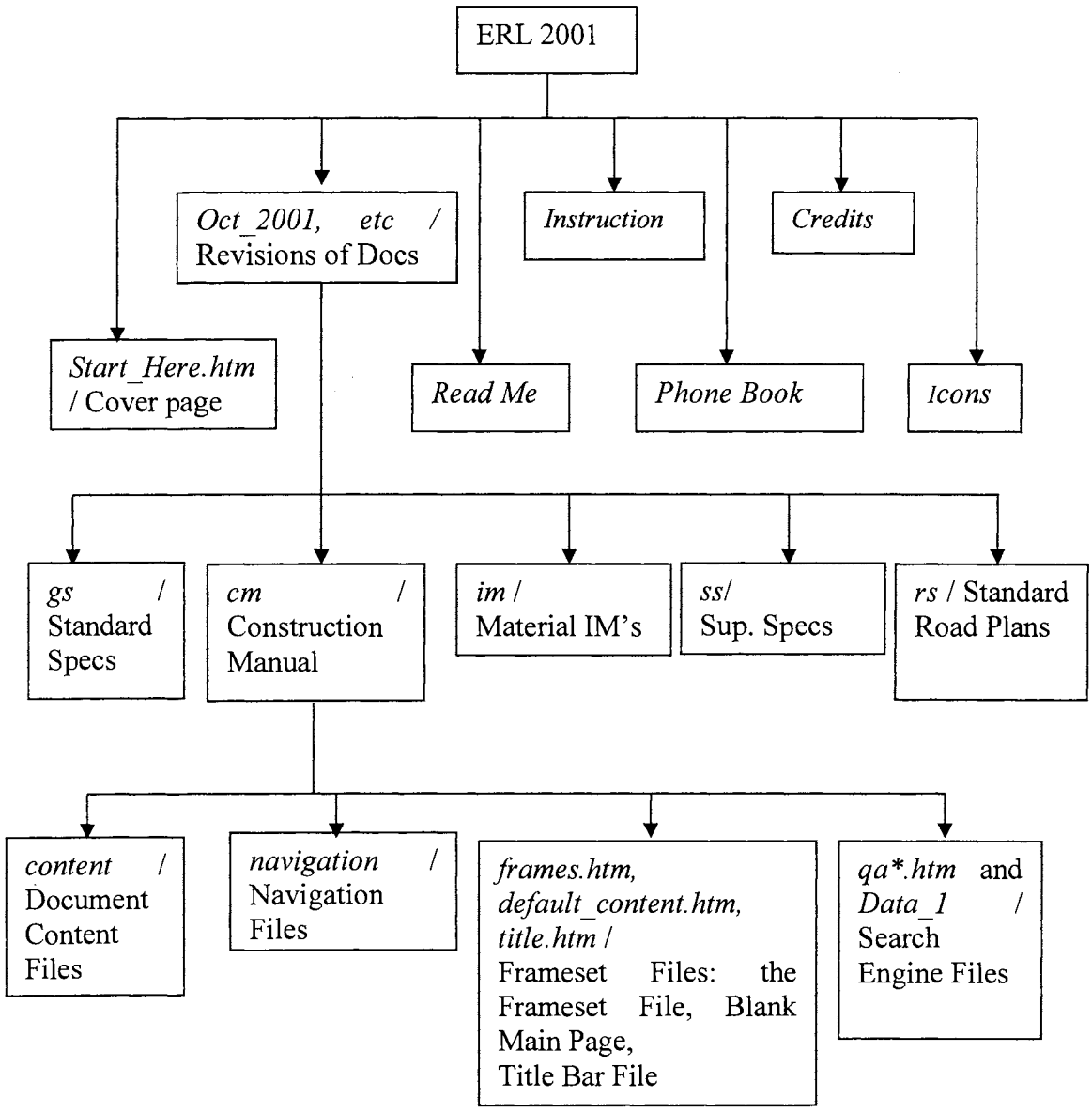


Figure 5-4 Standardized File Structure for the ERL

6. Search engine files: A search engine called "Quest Agent" is installed in the ERL for searching key words in the ERL documents. Search engine files are files used for the searching function. They are generated and named by a program called "Quest Agent Manager". These files include files named qa*.* and a directory

called data_1. They are now saved under the document directory such as "gs" or "cm".

7. Image files. All image files for icons used in the HTML files (especially in navigation files) are stored in a directory called "icons" which locates at the ERL root directory.
8. Avoid using space or punctuation as part of a file name.

The proposed file structure and naming convention solve the problems found in the original file structure. After using the new file structure, the gawk and *Perl* script codes for adding links or making other changes for one document can be easily modified for other documents because of the standardization in the file structures.

5.2 More Structured HTML

Introduction

As introduced in the previous chapter, the language with which many of the ERL documents are written, the HTML, has been changed considerably in recent years. The HTML codes of these documents were generated from Word/WordPerfect files using MS Word 97 or Corel WordPerfect 8.0 HTML converters. As these two programs were developed several years ago, they do not support the currrent version of HTML specification recommended by W3C. Some elements and attributes used in the ERL have been identified as "deprecated" or "obsolete" in HTML specification 4.01, which means that they may not be supported by future versions of web browsers. This requires that the HTML document files in the ERL be rewritten to conform to the newer HTML specification at an appropriate time in the future.

Another reason that conformation to the HTML 4.01 is desired is because of the new features that will be brought about by the HTML 4.01. As introduced earlier, one of the many new improvements in HTML 4.01 is the full support of Cascading Style Sheets (CSS) which allows separation between the file structure and presentation. We can see the difference between earlier versions of ERL and the HTML 4.01 from the example taken from the *Construction Manual*.

Example:

To present the section title of 10.70, which appears as follows:

"10.70 WELLS AND WATER POLLUTION",

when written in earlier version of HTML, the code goes like this:

```
<B><P>10.70 WELLS AND WATER POLLUTION</P></B>
```

When written in HTML 4.01, the code will become:

```
<h1>10.70   WELLS AND WATER POLLUTION</h1>
```

with the following scripts included as part of the CSS:

```
h1
      {margin:0in;
      margin-bottom:.0001pt;
      page-break-after:avoid;
      font-size:11.0pt;
      font-family:Arial;}
```

In earlier versions of the HTML, the HTML code addresses the need to present the texts in the desired format. Instead of doing this, HTML 4.01 addresses more about the function of the texts as a part of the whole article. The element "H1" indicates that the texts enclosed by "<H1>" and "</H1>" serves as a first level heading in the article. And the code included in the CSS specifies a uniform format to all such "H1" headings.

In the ERL case, this may bring great convenience when we use *awk* scripts to automatically insert bookmarks and hyperlinks in the raw HTML codes generated by a word processor.

Using Styles in Word XP

Because the HTML codes of ERL document files are converted from Word/WordPerfect files, we need a Word processing program that convert documents to HTML 4.01. Microsoft Word XP is found to be an excellent tool for this task. To achieve the desired structure and succinctness in the converted HTML file, the Word files need to be formatted properly before the conversion. As an example of illustrating how to edit the word file properly and the actually effect of the HTML file, a section from the Construction Manual will be re-formatted in Word XP and converted to HTML. Section 7.60 (Figure 5-5) is selected here because it is short and has most of the formats to used in other *Construction Management* files.

The point of using Word XP properly here is to use the styles function properly when formatting the Word documents. Section 7.60 was originally compiled in Word 97 and styles are not used in the desired manner. Thus we need to reformat the document.

Before formatting a document with styles, an analysis of the document structure will be helpful. In the example of *Construction Manual* 7.60, five styles (Figure 5-5) are needed to format the main content of the file:

- H1: The first level of headings can be used here to name the style for the title of the section file, "7.60 …", in this case;
- H2: The second level of headings can be used here to name the style for the titles of the subsections, such as "7.61 …", etc;
- H3: The third level of headings can be used here to name the style for subtitles;
- Normal: Majority of the document contents can be formatted using the Normal style;
- Reference: A few references to other documents exist in the file. We can use a style called "reference" to mark these texts.

Now, we can start reformatting the file using styles in Word XP:

1. Open the file in Word XP;
2. Show the **Styles and Formatting** box (the **Styles** box) by clicking **Format – Styles and Formatting**. Clear existing formats by selecting all contents and pressing the "**Clear all format**" button in the **Styles** box. Now all formats are cleared and only a few styles remain in the **Styles** box.
3. Highlight the texts you want to format as "**Heading 1**". Click "Heading 1" in the **Styles** box. Modify the format of "**Heading 1**" by right clicking the style name in the **Styles** box; choose "**Modify**" in the drop-down menu; change the format to the desired appearance using the dialogue box that will appear.
4. Repeat step 3 to apply styles for other parts of the text. If a style name is not listed in the **Styles** box. Click the "New Style" button in the upper portion of the **Styles** box to create one.

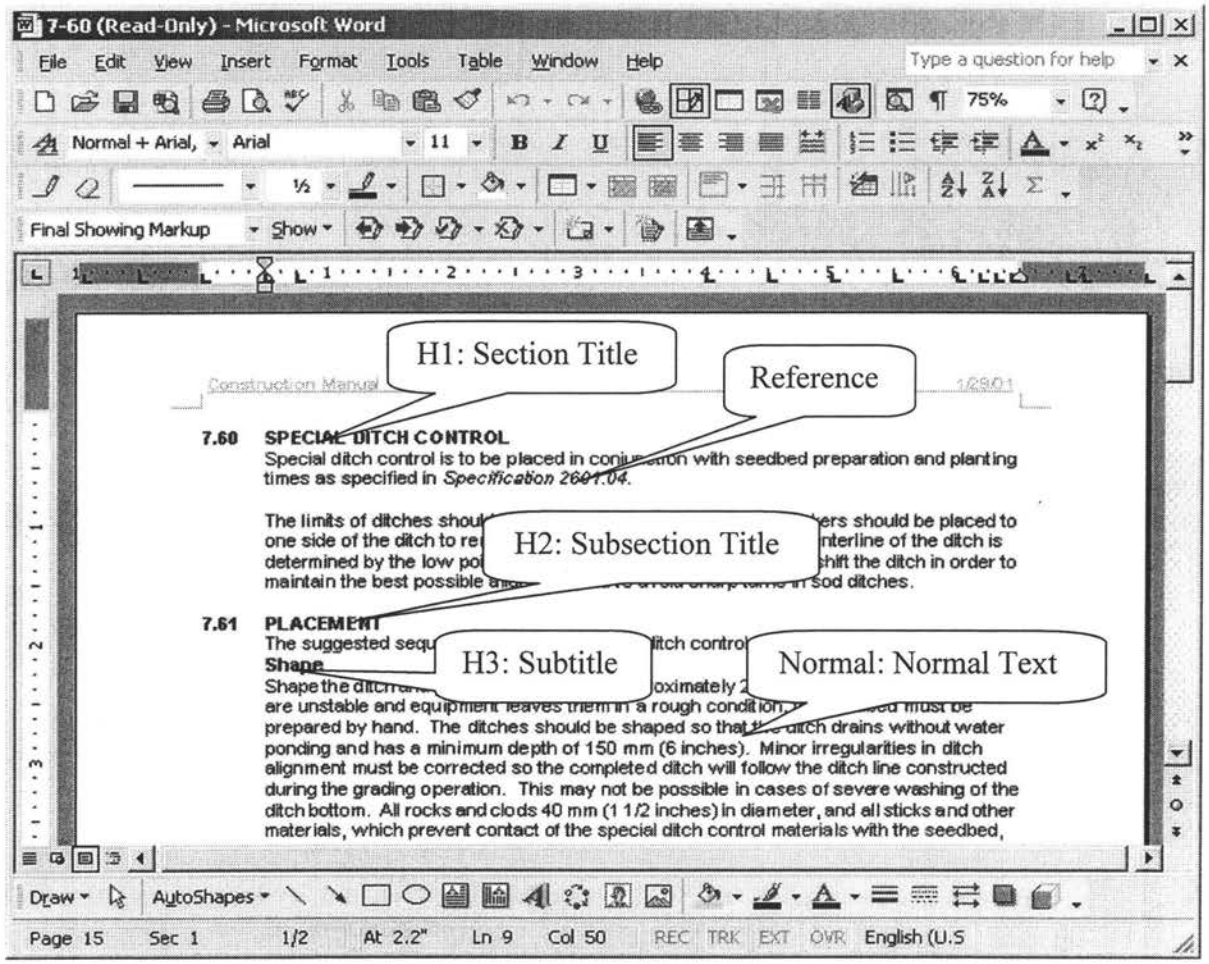


Figure 5-5 Styles used in Construction Manual 7.60

Now the reformat of the Word document is finished. As numbers in the various levels of titles must be preserved exactly as presented in the original documents when the *gawk* scripts are used, in the g*awk* scripts, automatic numbering is not desired when formatting either *Construction Manual* or the *Specifications* books. Using styles in editing word files brings convenience to the editing work. After applying the styles to the texts, you can change format of the texts by changing format of the styles instead of going through the document and making changes for each occurrence. If the styles to be used in a document can be determined in advance, you can edit the styles in an empty document and save the document as a template, which can be used for other documents with same file structure and format. If a CSS file is available, we can link the CSS file to the document to be formatted and apply the styles to document texts.

The Converted HTML Codes

After reformatting the Word document, we can convert it to a HTML file by simply saving it as a web page. To make the HTML code simple, brief and easy to understand, remember to select "Web page, filtered" in the **Save as type** box when saving the file. A part of the converted HTML code which conforms to HTML 4 is shown in Figure 5-6. The style sheet is shown in Figure 5-7. We can use *gawk* scripts to make further changes to the converted HTML files as described in Chapter 4.

```
<body lang=EN-US style='text-justify-trim:punctuation'>

<div class=Section1>

<p class=MsoNormal> </p>

<h1>7.60     SPECIAL DITCH CONTROL</h1>

<p class=MsoNormal>Special ditch control is to be placed in conjunction
with seedbed preparation and planting times as specified in <span
class=referenceChar>Specification 2601.04</span>.</p>

<p class=MsoNormal> </p>

<p class=MsoNormal>The limits of ditches should be staked with flats or
flags, and markers should be placed to one side of the ditch to remain
visible during construction.  The centerline of the ditch is determined
by the low point of the ditch.  It may be necessary to shift the ditch
in order to maintain the best possible alignment and to avoid sharp
turns in sod ditches.</p>

<p class=MsoNormal> </p>

<h2>7.61     PLACEMENT</h2>

<p class=MsoNormal>The suggested sequence of work for special ditch
control is as follows:</p>
```

Figure 5-6 HTML 4 Code for CM 7.60 Converted by Word XP

```
/* Style Definitions */
 p.MsoNormal, li.MsoNormal, div.MsoNormal
      {margin-top:0in;
      margin-right:0in;
      margin-bottom:0in;
      margin-left:.5in;
      margin-bottom:.0001pt;
      text-autospace:none;
      font-size:11.0pt;
      font-family:Arial;}
h1
      {margin:0in;
      margin-bottom:.0001pt;
      page-break-after:avoid;
      text-autospace:none;
      font-size:11.0pt;
      font-family:Arial;}
h2
      {margin:0in;
      margin-bottom:.0001pt;
      page-break-after:avoid;
      text-autospace:none;
      font-size:11.0pt;
      font-family:Arial;}
h3
      {margin-top:0in;
      margin-right:0in;
      margin-bottom:0in;
      margin-left:.5in;
      margin-bottom:.0001pt;
      page-break-after:avoid;
      text-autospace:none;
      font-size:11.0pt;
      font-family:Arial;}
p.MsoHeader, li.MsoHeader, div.MsoHeader
      {margin:0in;
      margin-bottom:.0001pt;
      text-autospace:none;
      font-size:11.0pt;
      font-family:Arial;}
span.referenceChar
      {font-family:SimSun;
      font-style:italic;}
 /* Page Definitions */
 @page Section1
      {size:8.5in 11.0in;
      margin:.8in 1.0in .8in 1.0in;}
div.Section1
      {page:Section1;}
```

Figure 5-7 The CSS codes for CM 7.60

Summary

In a way switching to HTML 4 or later is an inevitable change that must eventually occur for the ERL. The best timing would be when a new version of Construction Manual or Standard Specifications is published. Some changes in the practices of editing such contract documents at the IDOT would be desirable for the work of ERL development. These changes include converging to one word processing program and use the editing method that leads to desired HTML scripts, as is the method described in this section.

Before actually switching to HTML 4 in the ERL, an investigation on the versions of browsers used by target users is necessary. Compatibility of such browsers with HTML 4 needs to be studied before using the new version of HTML in the ERL.

CHAPTER 6. CONCLUSIONS

The Electronic Reference Library project at Iowa DOT and Iowa State University aims to develop and maintain a hypertext electronic publication that contains IDOT standard contract documents. To include the many changes to the contract documents in the ERL, a new version of the ERL is released every six months. So far a few versions of it have been issued and have been given favorable evaluations by the users. While the ERL brings convenience and efficiency to the target users, implementation and maintenance of the ERL had been a painful experience for the project members.

It is the objective of this thesis to find ways to improve the maintainability of the ERL, and provide efficient and effective tools and methods for ERL implementation and maintenance. To do this, challenges to the ERL implementation and maintenance work were first identified based on the author's experience. Then a review of the work of earlier researchers for the ERL project was conducted to understand users' need and the solutions proposed by earlier researchers. Thirdly, a review of relevant technologies was made to search possible improvements in the ERL and the implementation and maintenance methods. Based on these reviews and the experience of implementing and maintaining the ERL, a few tools and methods were proposed to address the various challenges faced by the ERL project team and two ways of improving the maintainability of the ERL are proposed.

Preparation (formatting, adding bookmarks and hyperlinks) of HTML files and checking errors in the PDF files are two of the most time consuming works in the ERL maintenance. To improve the work efficiency, a method of using *gawk* and *Perl* scripts to automate the preparation and error checking was used in the working process and documented in this thesis. To make the scripting easier, a few changes were suggested to be made to the ERL file storage structure and the HTML codes. Documentation of the working procedures and a PSWS were used in the ERL project to improve communication and share of knowledge. A few methods of collecting users' feedback were also discussed because user input has always been a major impetus and source for improving the ERL.

Most of the methods proposed in the thesis have been used in the project or at least tested in practical cases. These methods have been found significantly helpful in improving

work efficiency and quality. Constrained by the time and certain conditions, the surveying methods have not had a chance to be practiced in the ERL project and their effects remain uncertain. These methods are found potentially helpful from theoretical reasoning and should be practiced whenever possible.

The ERL project will continue for the foreseeable future. Because of the changes in users' needs, available tools, etc, searching for improvements in the ERL and its development methods will be a continuous work. Based on the researches documented in this thesis and the experiences of the author, a few recommendations for later ERL maintainers are listed below:

- Continue the effort of including more desired documents and links in the ERL to meet users' needs.
- Collect feedbacks from users.
- Search for the necessity and possibility of using new forms of storage and distribution method.
- Develop ERL for the PDA's and other handheld devices.
- Use video and audio files to illustrate testing methods, etc, in the ERL.

APPENDIX A. SHELL SCRIPT FOR PROCESSING *STANDARD SPECIFICATIONS* HTML FILES

```
#!/bin/bash

#[3]Pre-format
#The following part of gawk script pre-formats the Division HTML files,
#delete redundant and unwanted formats
gawk '{
      gsub(/>[ \t]*</, "><", $0);
      gsub(/<BODY[ A-Za-z0-9=#\"]*>/, "<BODY>", $0);
      gsub(/<BR [WP=\"BR0-9\"]+>/, "", $0);
      gsub("<FONT[ A-Za-z0-9=\"\-]*>", "", $0);
      gsub("</FONT>", "", $0);
      gsub("<STRONG></STRONG>", "", $0);
      gsub(/ /, " ", $0);
      if ($0 !~/[^:blank:]/) {$0=""}
      if ($0!="") {print $0 > FILENAME"1";}
      }' Div_*.htm

#Following are Linux commands that delete previous process results.
rm -f content/*.*
rmdir content
mkdir content

#Head
#The following gawk script identifies name for each section in #the
Division HTML file, generate file name for each section, and
#generate the HEAD part of the HTML codes.
gawk '{
#Initialize variable value.
      if (FNR == 1) {Title="";FileName="";}

#Identify and generate filename and <head>...</head> part for ##00.htm
      if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/DIVISION/) {
            for (i=1; i<=NF; i++) {
                  if ($i~/DIVISION/)      {
                              Divn=substr($(i+1),1,2);
                              SN=Divn"00";
                                    }
                              }
            FileName="./content/"SN".htm";
            gsub(/<P>/, "", $0);
            gsub(/<CENTER>/, "", $0);
            gsub(/<STRONG>/, "", $0);
            gsub(/<\/STRONG><\/CENTER>/, "", $0);
            Title=$0;
            print "<HTML>" > FileName;
            print "<HEAD>" > FileName;
            print "<META HTTP-EQUIV=\"Content-Type\" CONTENT=\"text\/html;
charset=windows-1252\">" > FileName;
```

[3] # indicates that this line is a line of comment.

```
            print  "<META  NAME=\"Generator\"  CONTENT=\"Corel  WordPerfect
8\">" > FileName;
            print "<TITLE>"Title"<\/TITLE>" > FileName;
            print "</HEAD>" > FileName;
            print "<BODY>" > FileName;
            print "<Font FACE=\"Courier New\" SIZE=\"2\">" > FileName;
                                                        }

#Identify and generate filename and <head>...</head> part for ####.htm
      if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/Section/) {
            for (i=1; i<=NF; i++) {
                  if ($i~/Section/) {SN=substr($(i+1),1,4);}
                                    }
            FileName="./content/"SN".htm";
            gsub(/<P><CENTER><STRONG>/, "", $0);
            gsub(/<\/STRONG><\/CENTER>/, "", $0);
            Title=$0;
            print "<HTML>" > FileName;
            print "<HEAD>" > FileName;
            print "<META HTTP-EQUIV=\"Content-Type\" CONTENT=\"text\/html;
charset=windows-1252\">" > FileName;
            print  "<META  NAME=\"Generator\"  CONTENT=\"Corel  WordPerfect
8\">" > FileName;
            print "<TITLE>"Title"<\/TITLE>" > FileName;
            print "</HEAD>" > FileName;
            print "<BODY>" > FileName;
            print "<Font FACE=\"Courier New\" SIZE=\"2\">" > FileName;
                                                              }
      }' Div_*.html

#Body
#The following gawk script generates the BODY part of the HTML file for
#each section with bookmarks and hyperlinks inserted.

gawk '{
#if (FNR == 1) {Title=""; FileName=""; SN=""; ssn="";}
if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/DIVISION/) {
            for (i=1; i<=NF; i++) {
                  if ($i~/DIVISION/)      {
                              Divn=substr($(i+1),1,2);
                              SN=Divn"00";
                                    }
                              }
      FileName="./content/"SN".htm";
                                                        }

if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/Section/) {
            for (i=1; i<=NF; i++) {
                  if ($i~/Section/) SN=substr($(i+1),1,4);
                                    }
      FileName="./content/"SN".htm";
                                                        }


#Bookmark a Section (Section ####) Title
```

```
if  ($0~/^<P><CENTER><STRONG>[  \t]*Section[  \t]*[0-9][0-9][0-9][0-9]\.[0-
9]*[ \t]+[A-Z]*/)       {
      i=match($0, /[0-9][0-9][0-9][0-9]\./);
      ssn=substr($0,i,4);
      name=ssn;
      sub(/<STRONG>/, "<STRONG><a name=\""name"\">", $0);
      sub(/<\/STRONG>/,"<\/a><\/STRONG>", $0);
      }

#Bookmark a Subsection (####.##) Title
if ($0~/^<P><STRONG>[0-9][0-9][0-9][0-9]\.[0-9][0-9][ \t\.]+[A-Z]*/)    {
      i=match($0, /[0-9][0-9][0-9][0-9]\.[0-9][0-9]/);
      ssn=substr($0,i,7);
      name=ssn;
      sub(/<STRONG>/, "<STRONG><a name=\""name"\">", $0);
      sub(/<\/STRONG>/,"<\/a><\/STRONG>", $0);
      }

#Bookmark a SubTitle.  The following patterns are considered a subtitle:
"<P><STRONG>A. AAKFK....".
#If  this  pattern  appears  in  subsection  ####.##,  it  is  bookmarked
"####.##A".
      if ($0~/<P><STRONG>[ \t]*[A-Z]\.[ \t]/)   {
            for (i=1; i<=NF; i++)   {
                  if ($i~/[A-Z]\./)       {
                        subttl=substr($i,(length($i)-1),1);
                        name=ssn""subttl;
                                    }
                              }
            sub(/<STRONG>/, "<STRONG><a name=\""name"\">", $0);
            sub(/<\/STRONG>/,"<\/a><\/STRONG>", $0);
                                    }

#Add links to ##00.htm.
if (FileName ~ /[1-9][0-9]00\.htm/) {
      if ($0~/<P><STRONG>[ \t]*[0-9][0-9][0-9][0-9]\.[ \t]+/)     {
            mat=match($0,/[0-9][0-9][0-9][0-9]/);
            flname=substr($0, mat, 4);
            sub(flname, "<a href=\""flname"\.htm\">"flname"<\/a>", $0);
                                          }
                              }

# Make links to strings with "Section[s]/Article[s] ####.[##]"
for (i=1;i<= NF; i++)    {

      if  (($i  ~  /Article[s]*/)  ||  ($i~/Section[s]*/)  )  &&  ($0  !~
/<CENTER>/ )) { SectSwitch = "T"}

      if
(($i!~/Article[s]*/)&&($i!~/Section[s]*/)&&($i!~/Paragraph/)&&($i!~/[A-
Z][,\.]*/)&&($i!~/and/)&&($i!~/[0-9][0-9][0-9][0-9]/))
            {SectSwitch="F"}

        if ((SectSwitch=="T") && ($i~/[0-9][0-9][0-9][0-9]/))     {
```

```
            if ($i~/[0-9][0-9][0-9][0-9]\.[0-9][0-9]/)            {
                  subSectSwitch="T";
                  mat=match($i,/[0-9][0-9][0-9][0-9]\.[0-9][0-9]/);
                  flname=substr($i,mat,4);
                  ssname=substr($i,mat,7);
                  sub(ssname,                                              "<a
href=\""flname"\.htm#"ssname"\">"ssname"<\/a>", $i);

                                                        }
            else              {

                  if ($i~/[0-9][0-9][0-9][0-9]/)            {
                        mat=match($i,/[0-9][0-9][0-9][0-9]/);
                        flname=substr($i,mat,4);
                        sub(flname,                                        "<a
href=\""flname"\.htm\">"flname"<\/a>", $i);
                                                  }

                              }
                                                  }

      if                (($i!~/[0-9][0-9][0-9][0-9]\.[0-9][0-9]/)&&($i!~/[A-
Z][,\.]*/)&&($i!~/and/))
            {subSectSwitch="F"}

      if ((subSectSwitch=="T") && ($i~/^[A-Z][,\.]*/) && ($i!~/Para/))  {
                        mat = match ($i, /[A-Z]/);
                        sttlname=substr($i,mat,1);
                        sttlbkmk=(ssname sttlname);
                        sub(sttlname,                                      "<a
href=\""flname"\.htm#"sttlbkmk"\">"sttlname"<\/a>", $i);
                                                  }
                  }


# Make links to Materials I.M.#
for (i=1;i<= NF; i++)    {

      if (($i ~ /[Mm]aterials/) && ($(i+1) ~/I.M./)) { MtrlSwitch ="T" }

      if (($i !~ /[Mm]aterials/) && ($i !~/I.M./) && ($i !~ /[0-9][0-9][0-
9]/)  &&  ($i!~/Appendi/)  &&  ($i  !~/and/)  &&  ($i  !~  /^[A-Z][$,\.]/))
{MtrlSwitch ="F"}

      if ((MtrlSwitch == "T") && ($i ~/[0-9][0-9][0-9]/)) {

            mat=match($i, /[0-9][0-9][0-9]/);

            if ($i ~ /[0-9][0-9][0-9]\.[0-9][0-9]/) {

                  IMname=substr($i, mat, 6);

                  if (($(i+1) ~ /Appendi/) && ( $(i+2) ~ /^[A-Z]/)) {
                        app=substr($(i+2), 1, 1)
                        IMsflname=(IMflname"a"app);
```

```
                        sub(IMname,                                                 "<a
href=\"..\/..\/Im\/"IMsflname".pdf\">"IMname"<\/a>", $i);
                                                                                    }
                    else  {                     sub(IMname,                         "<a
href=\"..\/..\/Im\/"IMname".pdf\">"IMname"<\/a>", $i);}

                                                }

            else  {

                    IMname=substr($i, mat, 3);

                    if (($(i+1)~/Appendi/) && ($(i+2) ~ /^[A-Z]/)) {

                          App=substr($(i+2), 1, 1);
                          app=tolower(App);
                          IMsflname=(IMflname"a"app);
                          sub(IMname,                                               "<a
href=\"..\/..\/Im\/"IMsflname".pdf\">"IMname"<\/a>", $i);
                                                                                    }
                    else  {                     sub(IMname,                         "<a
href=\"..\/..\/Im\/"IMname".pdf\">"IMname"<\/a>", $i);}

                    }

                                                }

                    }
if ($0 ~ /<[Bb][Oo][Dd][Yy]>/) {startWrite="T"}
if ($0~ /<\/[Bb][Oo][Dd][Yy]>/) {startWrite="F"}
if    ((startWrite    ==    "T")&&($0!~/<[Bb][Oo][Dd][Yy]>/)&&(FileName!=""))
{print $0 >>FileName;}
}' Div_*.html

#End
#The following gawk script generates endings to the HTML files
gawk '{
if (FNR==1) {     print "</FONT>" >> FILENAME;
      print "</BODY>" >> FILENAME;
      print "</HTML>" >> FILENAME;
      }
}' content/*.htm
```

APPENDIX B.

Instruction Manual

on

Developing and Updating the Electronic Reference Library

By

Lifeng Li

Russell Walters

Kiran Ankanahalli-Shivaramu

Department of Civil and Construction Engineering

Iowa State University

Dec 9, 2001

Table of Contents

Table of Figures

Acknowledgements

Special thanks go to Tom Reis and Donna Buchwald, for their initiating the idea of developing this manual. Josh Kohlhaas and Patricia Magoon provided valuable comments and suggestions in the preparation process of this manual.

Credit should also be given to Dr. Charles Jahren, Chun Li, Ozdemir Cetin and other project members for their contribution in the ERL development.

Chapter 1. Introduction

This manual has been developed to delineate the composition of the Electronic Reference Library (ERL) for Iowa Department of Transportation and instructions on updating and maintaining the ERL.

The ERL is a combination of the electronic version of various construction documents currently used by Iowa Department of Transportation (IDOT). When this manual is prepared, five documents are included in the ERL. They are: *Standard Specifications with revisions incorporated, Supplemental Specifications (ss)*, *Construction Manual (cm)*, *Standard Road Plans (RS)*, and *Material Instructional Memorandums (IM)*. As one or more of these documents are to be updated every six months, an update is also to be made to the ERL every six months. The purpose of this Instruction Manual is to help successive updaters of the ERL to better understand:

1. How the ERL is organized;
2. What are the hardware and software requirements for ERL updates;
3. What technologies and skills are needed in the updates;
4. How to update the ERL efficiently;
5. How to find and minimize errors;

Although many issues and questions have been discussed and answered in the design and development stage, and need not be reconsidered in the update process, the update and maintenance of the ERL is anything but a simple task. To successfully get the ERL updated in time and eliminate errors, the updaters must know or learn how to use a number of software packages or utilities, and in certain cases may be required to do some programming work.

Software applications to be used in the update process are:

1. MS Word (97 or 2000);
2. Corel WordPerfect 8.0 or above;
3. MS FrontPage 2000;
4. Adobe Acrobat 4.0 or above;
5. Adobe Photoshop 5.5 or above;

6. Quest Agent 5.0 Pro;

Software packages suggested to be used include:

7. Gawk
8. Perl
9. Cygwin, which includes gawk and Perl.

Required programming skill includes:

10. HTML;

Suggested programming skills include:

11. JavaScript;
12. Gawk scripting;

It is not an easy task for a person to be able to master so many computer applications and skills, especially some of which are not so commonly used. And it must be noted that the purpose of this manual is not to teach the readers how to use any of the above-mentioned software tools. Familiarity with MS Word, Corel WordPerfect, MS FrontPage, Adobe Acrobat, Adobe PhotoShop HTML programming are assumed with the updaters. Instructions on how to use Quest Agent 5.0 Pro can be found with the software's help web pages. Complete instructions on how to program for gawk are available free at: http://www.gnu.org/manual/gawk-3.0.3/gawk.html, and will not be covered in this manual.

Knowledge of JavaScript or Perl is optional. Many sources of learning such two programming language can be obtained from the Internet or a library. The names of some of these books are listed below:

1. *Perl 5 by Example*, David Medinets
2. *Perl in a Week*, David Till
3. *JavaScript in a Week,* Armen Danesh
4. *Using JavaScript,* Mark Reynolds with Jerry Honeycutt

Chapter 2. Overview of ERL Maintenance and the File Structure

2.1 The Maintenance Process for ERL

Earlier researchers provided a figure (see Fig. 1.) describing the process they used to update the ERL (Chun Li, 2000).

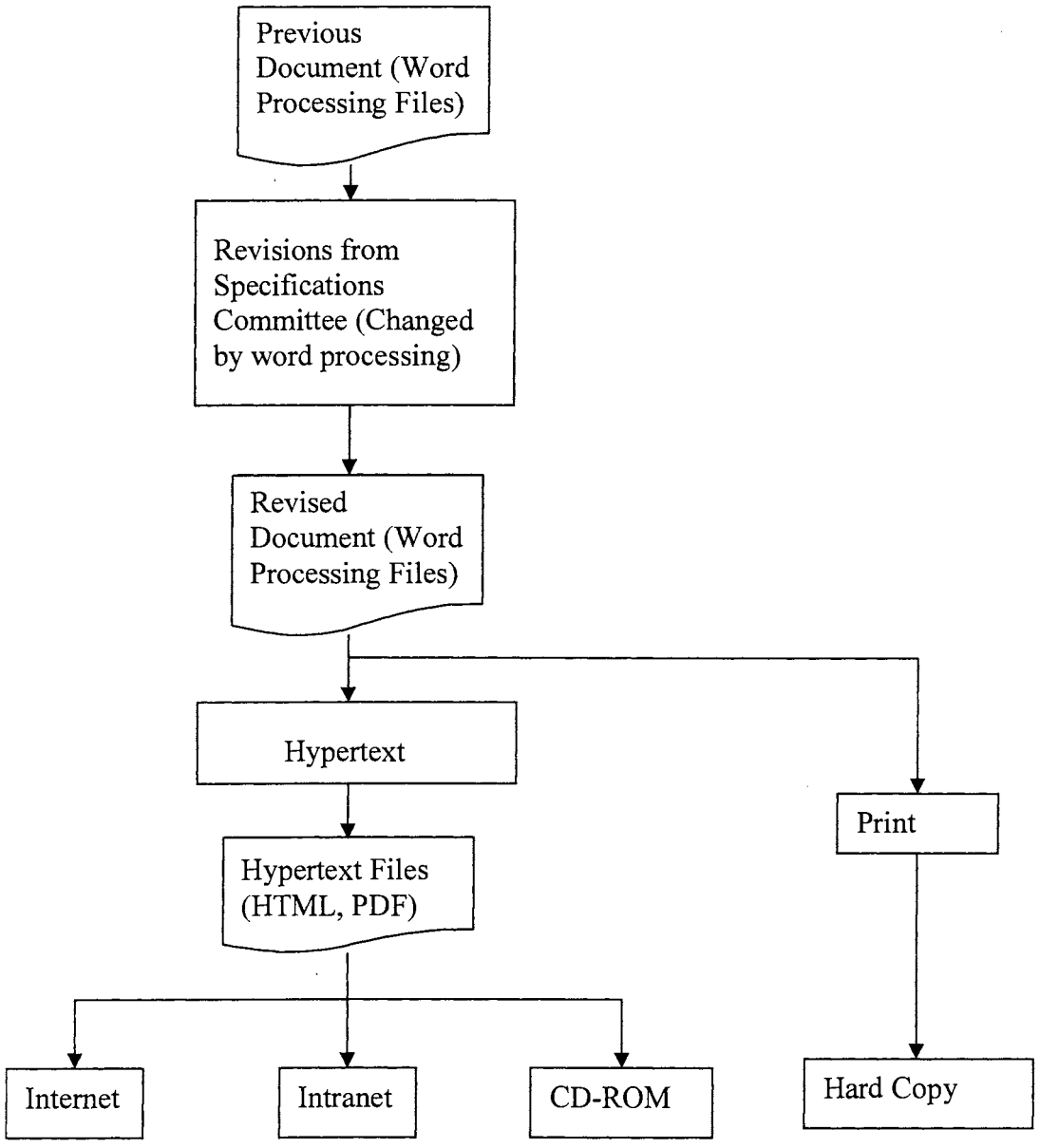


Fig. 1 The Maintenance Process Flowchart

Earlier researchers also suggested that after the ERL has been published, revisions to the documents could be made directly to the Hypertext Files.

2.2 Suggested File Storage Structure

The number of revisions included on a CD is determined by the storage limit of the CD. The suggested file storage structure is shown in Fig. 2.

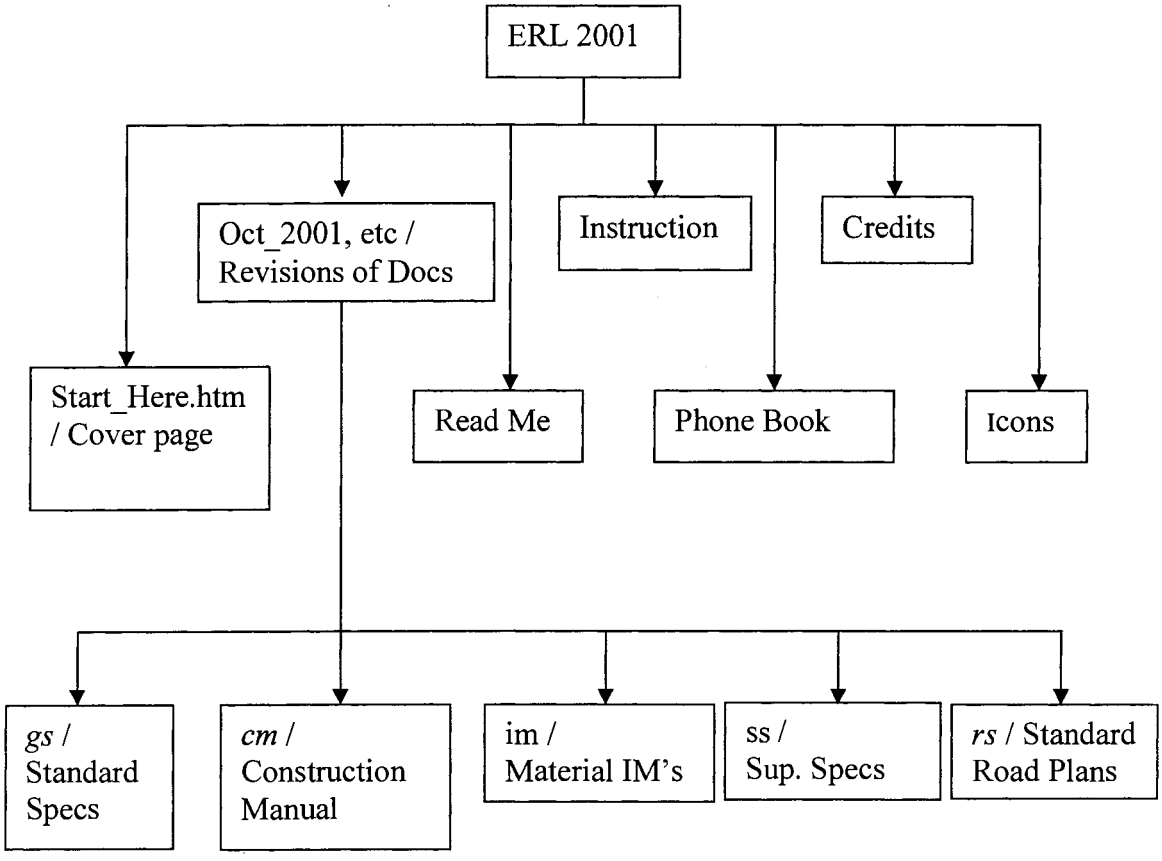


Fig. 2 First and Second Level File Structure for ERL

The functions of the files and folders at the root directory of the April 2001 version ERL are:

1. ***Start_here.htm***: This document is the opening or home page for the ERL, written in HTML.
2. ***Icons***: This folder contains images that are referred to in .htm files throughout the ERL CD.

3. ***Cover Page***: this folder contains files needed for the opening page, *Start_Here.htm*. The files are contained in two folders, *HM4* and *images*. *HM4* contains files for the hierarchical menu on the opening page. The *images* folder contains the images of the opening page. Instructions on how to update the hierarchical menu files and the opening page will be given in later chapters.
4. ***Phone Book***: This folder contains files for the Iowa Department of Transportation phonebook.
5. ***Credit Page***: This folder contains files needed for the credit page.
6. ***Instructions***: This folder contains files for the user's manual page.
7. ***Oct_2001***: This folder contains the documents valid for the letting date of October 2001. Name the directories after certain conventions, like *Oct_2001*.
8. ***Apr_2001***: This folder contains the documents valid for the letting date of April 2001.

As was said, the October 2001 letting has five documents, each contained in a directory under the directory named "*Oct_2001.*" The sixth directory in Oct_2001, "*scripts*", contains JavaScript files for the Quick jump function for the Oct_2001 documents.

Within the directory of each document, there are primarily four types of documents, either grouped in subdirectories or just as individual files(see Fig. 3.).

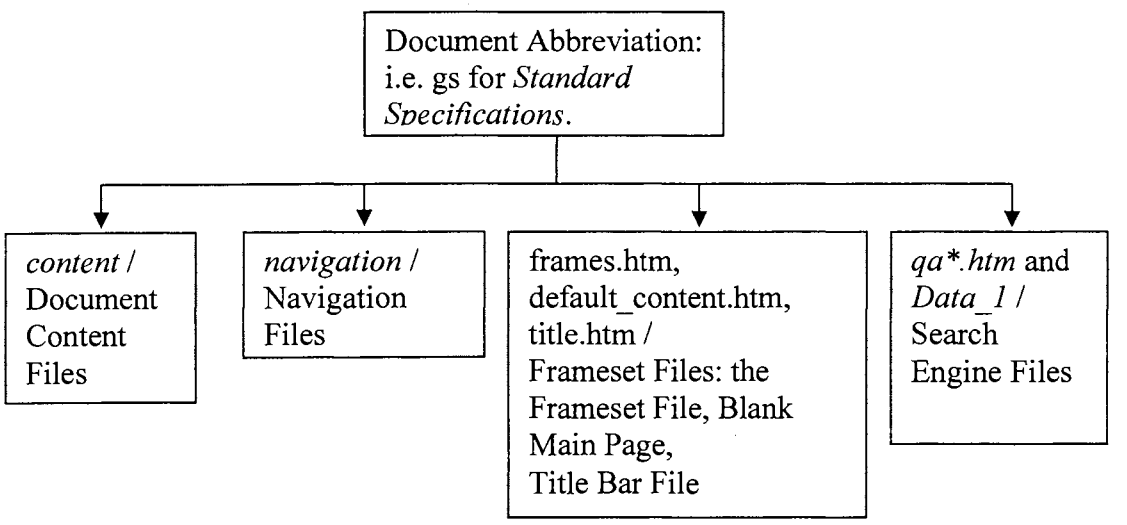


Fig. 3 Third Level File Structure for ERL

As an example, the file structure for gs is shown in Fig. 4.

Document content files: These files are files that actually contain the contents of the documents. Two types of files are used for document files, .htm (gs, ss, majority of cm) and .pdf (RS, IMs, and most Appendixes for CM.) Normally, each file contains one section of a chapter (in Construction Manual) or a division (in Standard Specifications), or a page in Road Standards or Materials Instruction Memorandums. It is suggested that these documents be stored in one directory, i.e. "content" as in the suggested template, to simplify the cross-links in the document files.

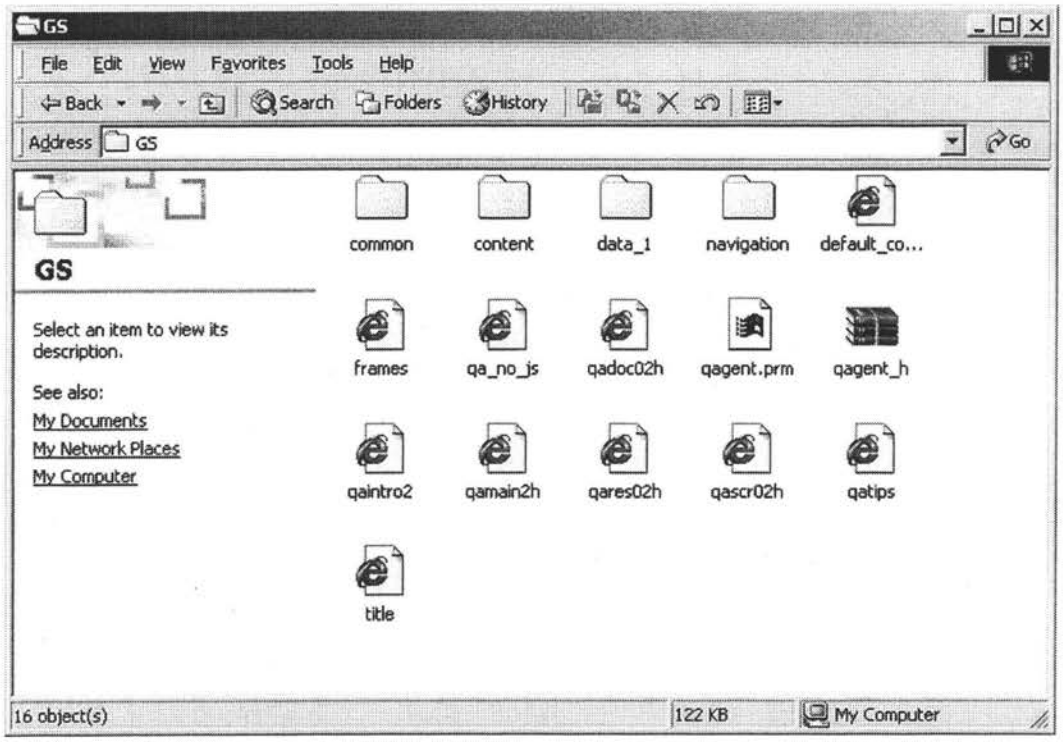


Fig. 4 Files and Subdirectories for Standard Specifications with General Supplemental Revisions for ERL 2001

Navigation files: are files that enable the user of the CD to reach the document files. They are similar to the *table of contents* in a book, and actually can be created conveniently on the basis of the *table of contents* of the documents. For most of the documents (Standard Specifications, Construction Manual, Road Standards), the navigation files consists of two levels. The first level lists the names of the chapters or divisions, and is the default content shown in the navigation bar (see Fig. 5.). The second level of navigation files lists the

contents of each chapter or division, and can be accessed by clicking the chapter name or division name at the first level of navigation file.

Frameset Files: It has been decided to use frames in ERL's graphical interface. Shown in Figure 5 is an example of the frameset used in ERL. The function of the file *"frames.htm"* in each document directory is to define the layout and naming of the frameset and which file is to be shown in each frame. There are three frames defined in the frameset, and are called "banner", "contents", "main" respectively. *"Title.htm"* defines the content to be shown in the "Title Bar" or "banner" frame. "D*efault_content.htm*" defines the default appearance of the "main" frame. Files in the subdirectory *"navigation"* will be displayed in the navigation bar frame or "contents" frame.

Note that the frame names are case sensitive. Use the frames exactly the way they are called when referring to these frames in HTML tags.

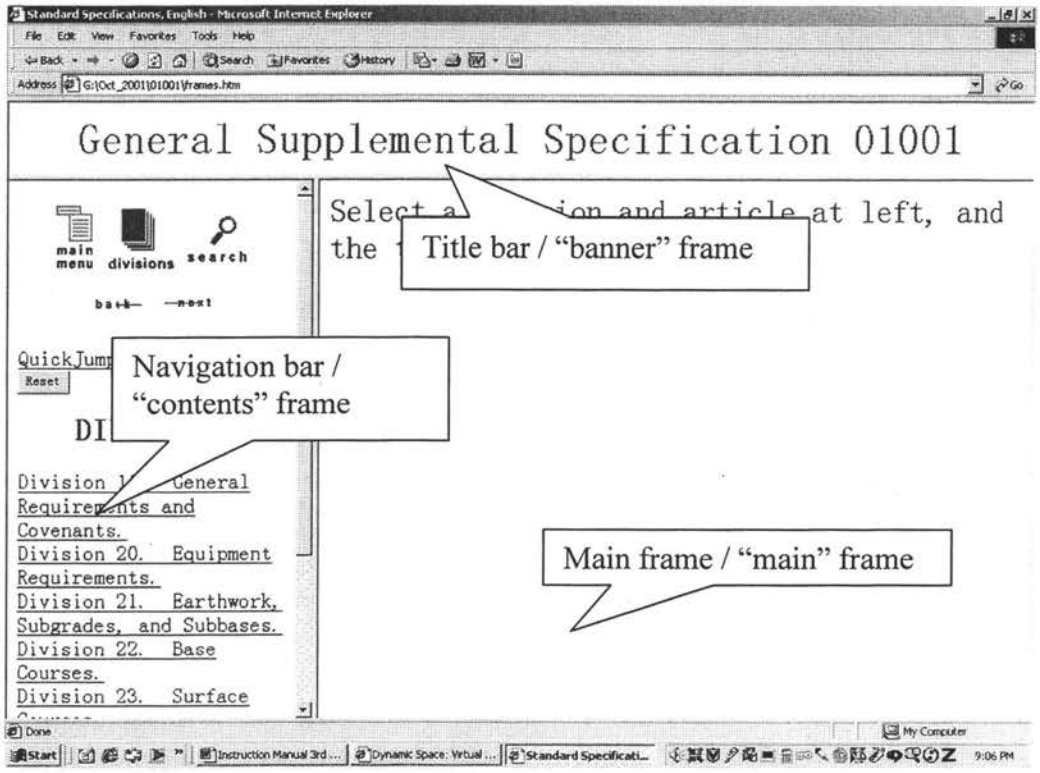


Fig. 5 The Opening Frameset for Standard Specifications

Search Engine Files: The search engine files are files with names "qa*.*," and the subdirectory "data_1." These files perform the function of searching certain key words or

phrases in a document. These search engine files are automatically generated and deployed by Quest Agent 5.0. More introductions on QuestAgent 5.0 will be given in Chapter 5.

Important: *The file structure should be carefully designed at the beginning of the update. Unless absolutely necessary the original file names and directory names should remain unchanged. Because the documents are cross-linked, any change of one directory or file name may cause or create invalid links in other files.*

Chapter 3. Preparing for Updating ERL

Before updating the ERL, we need to prepare for the update in two ways: meet the software and hardware requirements, and make a plan for your update.

3.1 Software and Hardware Requirements

Software and hardware requirements include:

1. Operating System: Windows NT 4.0, Windows 95/98 or higher.
2. Browser: Internet Explorer 4.0 or higher; latest versions of browsers are recommend. This Electronic Reference Library has limited functionality under Netscape Navigator (see "Known Problems and Limitations" in the ERL CD). Functionality limitations are mitigated when Netscape Navigator 4.72 or higher is used.
3. CD-RW, CD-ROM Drive.
4. Hard Disk: The ERL can be copied onto your hard disk and browsed from there. If this is done, 1 GB of hard disk space is required in addition to hard drive space requirements of the Operating System and applications.
5. Adobe Acrobat 4.0 or higher.
6. To use the search engine properly when accessing the ERL through Internet, the browser machine also need to install the latest version of Virtual Machine (VM). To update the VM on your machine, please go to: http://www.microsoft.com/java/.
7. MS Word 97/2000 or above
8. WordPerfect 8.0 or above
9. MS FrontPage 2000 or above
10. Adobe Photoshop 5.5 or above
11. Quest Agent 5.0 Pro

Software utilities suggested to be used are:

13. Gawk for DOS, or
14. Cygwin, which includes gawk

3.2 Prior Planning for the Update Process

Each update of the ERL can be considered as a small project. It has clear and pre-defined objectives and very strict time and cost restrictions. Good planning before the beginning of the actual update operation is very helpful and can be considered essential to a successful update. In this stage you need to find answers to the following questions:

1. What new documents are to be added to the new ERL? For each of the new documents, in what format are the source files stored? When and from whom you can get these source files? What format should be used for them in the ERL? How are you going to process these source files? How long will each activity take?
2. What documents are to remain unchanged and what are to be changed? What changes will be done to these documents? How would you incorporate such changes in the new ERL? Write down a detailed list of what activities you need to do to update the documents.
3. Write a detailed and complete list of activities that you need to do in updating the Cover Page, the Read Me, Phone Book, Credits, and User's Guide page.
4. How many people will be available for the ERL update? How much time can they devote to the update and what relevant skills do they have?

To determine the answers to the above question, a meeting is suggested and all involved parties should participate. The meeting should be held as early as possible and meeting minutes should be issued which record all decisions reached in the meeting. A master plan, which contains answers to the above questions, should be prepared and a schedule should be included in the master plan. The master plan should be reviewed and expressly agreed by all involved individuals.

Chapter 4. Creating Document Files

4.1 Document File Creation Process

Creating the .htm and .pdf format of the document files that will eventually appear in the final ERL is the major part of the ERL update work. The creation process is illustrated in the flowchart diagram (Fig. 6).

4.2 File Format Selection

Two file formats, PDF (Adobe Portable Document Format) and HTML (HyperText Markup Language), are selected for the document files. HTML files are used mainly for text files and PDF format are mainly used for drawings, complicated tables and pages that are used for printing.

4.3 Creating HTML Files

A. Converting Word/WordPerfect files to HTML format

The input files for making HTML format of document files provided by the Specification Section of Iowa DOT are either edited with MS Word or Corel WordPerfect. The purpose of this step is to create a draft of the HTML format of the document file by converting existing Word/WordPerfect files. The converted HTML files will be further formatted and edited in later steps. According to the file format of the original document files, the files can be converted to HTML files either using the Word converter or Word Perfect converter.

MS Office 2000 and HTML Filter 2.0

For the converted HTML files, we have a few preferences:

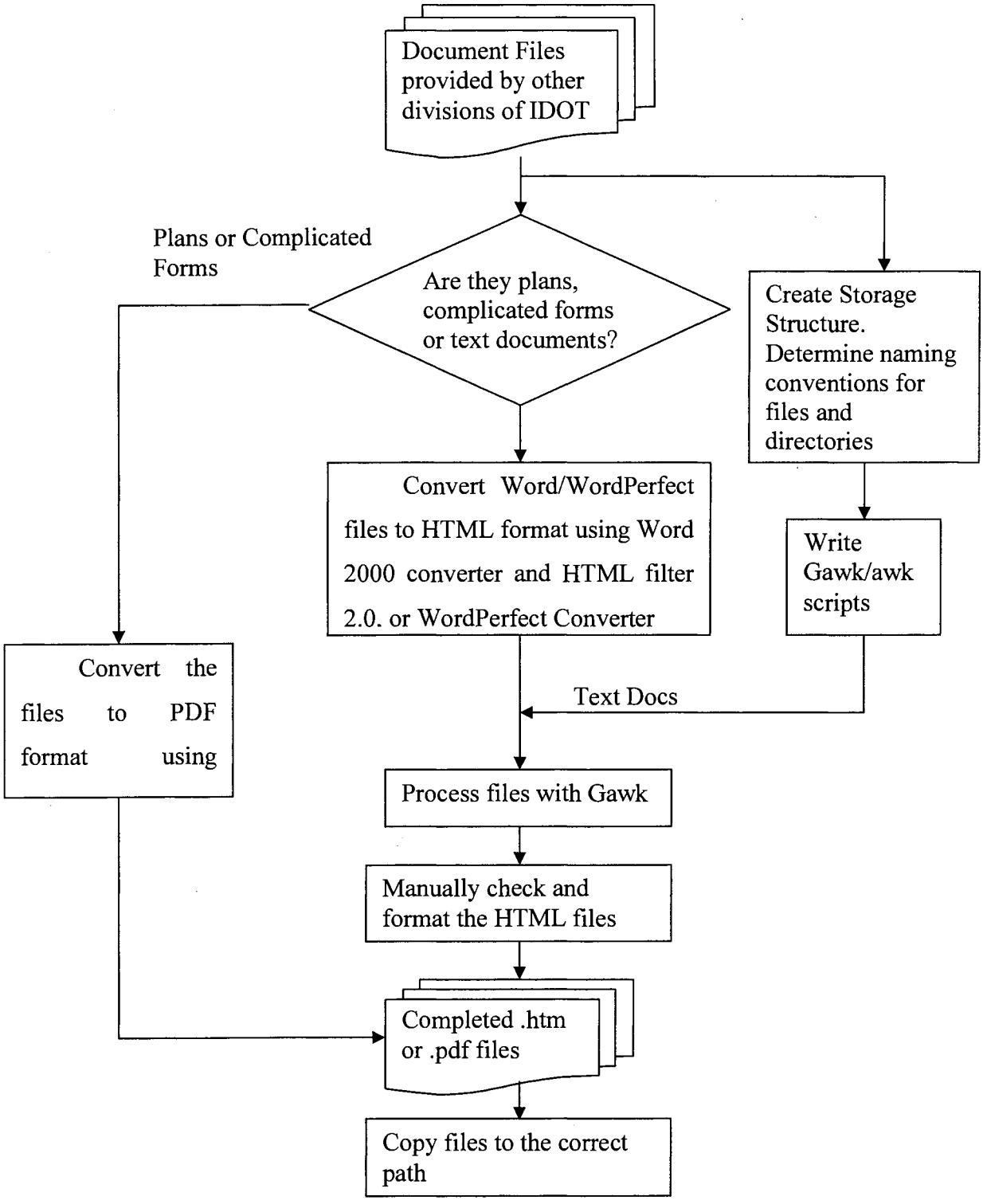


Fig. 6 Document file creation process

The Office-specific markup increases HTML file size. If you know you won't need to edit the HTML version of a document in Word again; you can trade round-trip capability for smaller file size by using Microsoft Office 2000 HTML Filter version 2.0. The filter removes Office-specific markup from HTML files created in Word so that they take up less storage space on Web servers and take less time for users to download. This process does not affect the appearance of your web pages.

For how to download, install and use Office HTML Filter version 2.0, please refer to the following web pages in Microsoft's web site:

a. http://office.microsoft.com/downloads/2000/Msohtmf2.aspx
b. http://office.microsoft.com/assistance/2000/htmlfilter.aspx
c. http://office.microsoft.com/assistance/2000/wDosPeeler.aspx
d. http://office.microsoft.com/assistance/2000/wMultiplePeeler.aspx

To convert the Word documents to HTML files using Word 2000 converter and Office HTML filter 2.0, you can follow the steps shown below:

a. Place the documents you want to convert in a single folder.
b. On the File menu, click New, and then click the Other Documents tab.
c. Double-click Batch Conversion Wizard (Fig. 7).

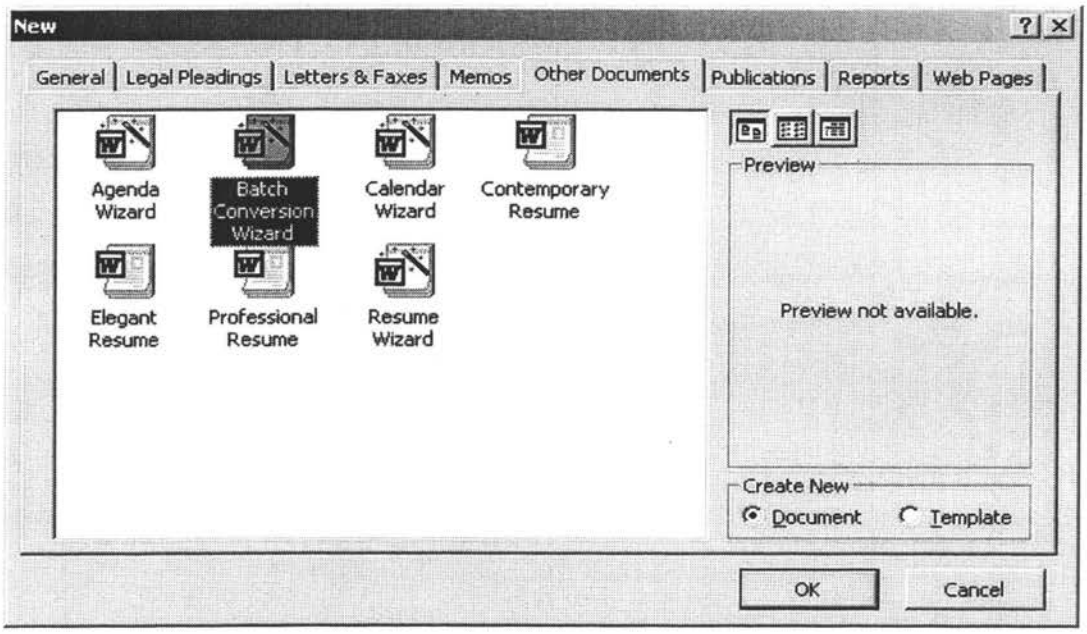


Fig. 7 Open the Batch Conversion Wizard

Note: If you do not see the Batch Conversion Wizard in the "New" dialog box, you may need to install it.

d. Follow the directions on the screen and save the result files in a single folder.

e. Click "Start – Programs – Microsoft Office Tools – Microsoft Office HTML Filter 2.0" from the Windows toolbar to open the MS Office HTML filter 2.0 (Fig. 8).

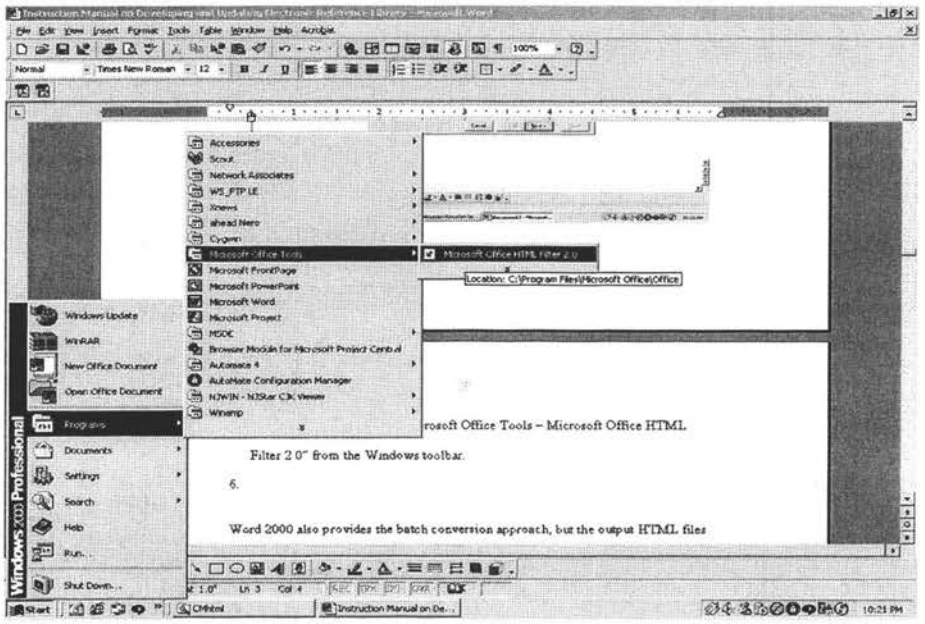


Fig. 8 Open Microsoft HTML Filter 2.0

f. Click "Add" to select the HTML files to be cleaned. And click Options to specify what format to be removed (Fig. 9).

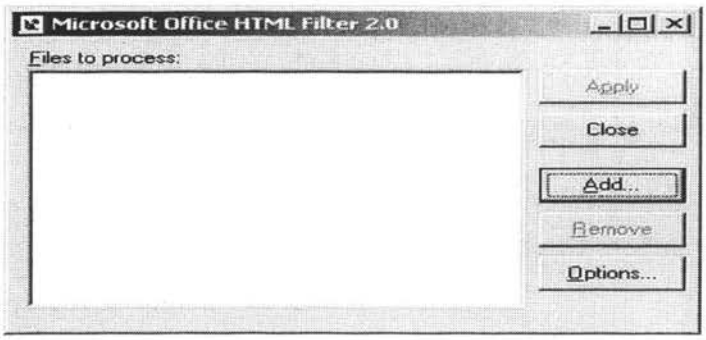


Fig. 9 Microsoft Office HTML Filter 2.0 Dialogue Box

The WordPerfect HTML converter

The key features of the WordPerfect HTML converter are similar to that of the Word 97. It supports batch conversions and the converted file is similar to that converted by Word 97, succinct and small in size. These features make WordPerfect a good choice for the conversion job.

From our experience of using WordPerfect to convert document files to HTML format, we found the following points are to be noted in the conversion process.

1. When using WordPerfect to convert Word documents, the word documents must first be converted to WordPerfect format. Fortunately, this can be done using the batch conversion function of WordPerfect and no major changes in the formats and contents were found between the files before and after the conversion.
2. When converting the WordPerfect files to HTML format, it is recommended that there are no "." in the file names of input files. For example, it is preferable to use "CM100.doc" than "CM 1.00.doc." Otherwise the output files may not be properly generated after the conversion.

To use the batch conversion function of WordPerfect,

1. Choose "Tools – Macro – Play" from the menu, by default the Play Macro – WPWin dialogue box appears.
2. In the dialogue box, select cvtdocs8 and click play.
3. Follow the instructions to finish the converting process.

B. Manually formatting the HTML files using MS FrontPage

After having converted the Word/WordPerfect files into HTML files using WordPerfect batch conversion function, we need to spend a little time to check the result of the conversion and make some corrections to the lost format. Two important corrections that should be made in this step are:

1. **Correct the section numbers.** For example, after the conversion by using the WordPerfect converter, *CM 10.10.doc* changes from the following appearance (Fig. 10):

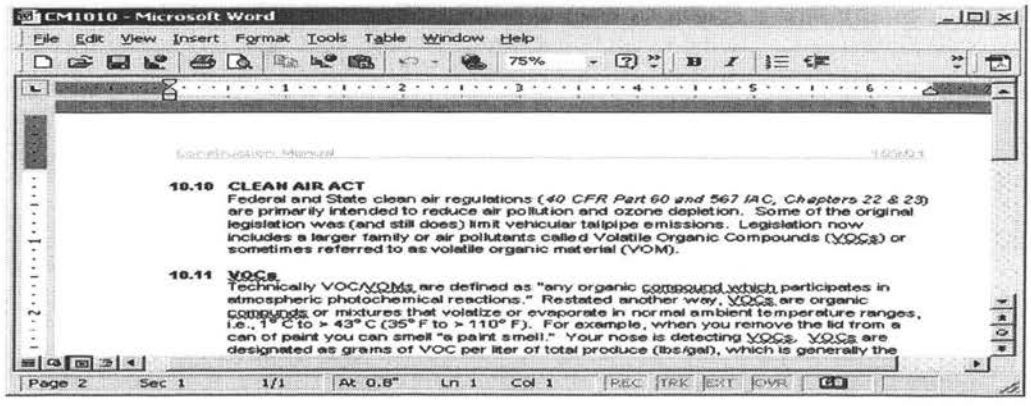


Fig. 10 Appearance of *CM 10.10.doc*

To (Fig. 11):

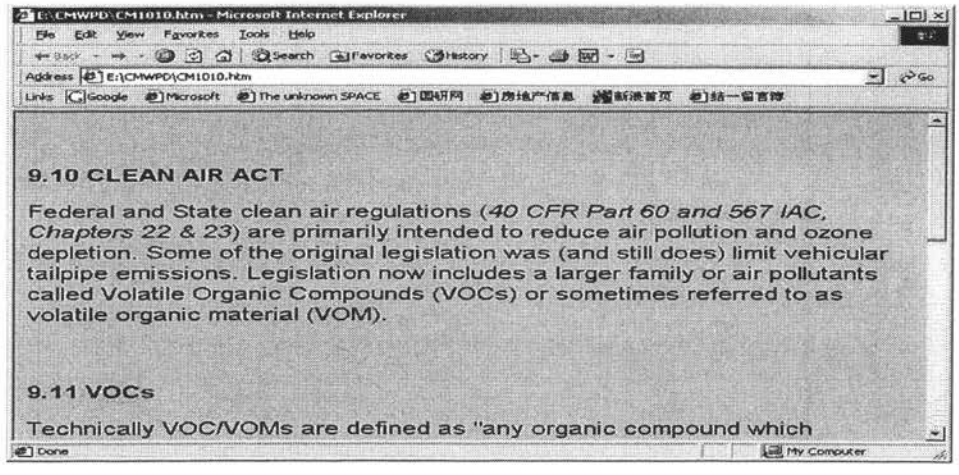


Fig. 11 Appearance of CM 10.10. doc after conversion by WordPerfect 8.0

Thus the subsections are incorrectly numbered and should be corrected.

2. Use tables instead of tab keys to format tables.

Many tables in the input files are formatted using the tab keys. After conversion, as there are no tags similar in function to the tab keys in HTML, the table format will be lost in the converted .htm files. In these cases, it is recommended that the updaters replace the format with tables using FrontPage or tables be used for these occasions in the original Word or WordPerfect files.

C. Edit the HTML files using gawk

After converting the document files and making sure that the HTML files sufficiently resemble their original counterparts, we can make further modifications to the HTML files. Four key types of modifications will be made to the HTML files:

1. Correct the Title of the HTML files.
2. Delete unnecessary format tags.
3. Add bookmarks to section names, subsection names, and subtitles.
4. Add links to the references.

Manually editing the document files are tedious, time consuming, and vulnerable to errors. A useful tool has been found to replace the tedious manual work. The name of the tool is awk.

Awk and Gawk

The term *awk* refers to a particular program originally used in the Unix operating system, and also to the language you use to tell this program what to do. When we need to be careful, we call the program "the *awk* utility" and the language "the *awk* language". The term *gawk* refers to a version of *awk* developed as part of the GNU project. Many computer users frequently need to make changes to text files wherever certain patterns appear, or extract data from parts of certain lines while discarding the rest. This type of work is easier with *awk*. And the above-described editing work with ERL happens to fall into such a category.

To learn about programming in the awk language, please refer to: http://www.gnu.org/manual/gawk-3.0.3/gawk.html.

Cygwin

Traditionally, *gawk* is a Unix program. Although there is *gawk* program for DOS, we are using an Environment called *Cygwin*. *Cygwin* is a UNIX environment emulator for Windows.

It consists of two parts:

- A DLL (cygwin1.dll), which acts as a UNIX emulation layer providing substantial UNIX API functionality.
- A collection of tools (including gawk), ported from UNIX, which provide UNIX/Linux look and feel.

The *Cygwin* DLL works with all versions of Windows since Windows 95, with the exception of Windows CE.

To install *Cygwin*, please go to: http://sources.redhat.com/cygwin/.

Programming with Gawk

In this section, we will talk about the syntaxes and functions of gawk scripts that will be used in the ERL updates. The following topics are selected from *Effective Awk Programming, A User's Guide for GNU Awk, Edition 1.0.3, February 1997*(Arnold D. Robbins). For people who want to learn the syntaxes and functions necessary in ERL updates, the topic list can serve as shortcut guidance.

Relevant topics include:

- Introduction
  - Typographical Conventions
- Getting Started with Awk
  - How to Run Awk Programs
  - When to Use Gawk
- Regular Expressions
- Reading Input Files
- Printing Output
- Expressions
- Patterns and Actions
- Control Statements
- Built-in Variables: FNR, NF, NR, FILENAME, RS, FS.

- Built-in Functions
  - Built-in Functions for String Manipulation

Using *Gawk* to Do the Conversion

After learning the basics of using *gawk*, we can start working on programming for the ERL update. This section introduces the methodology and logics of developing sample gawk scripts, which are intended to automatically process the HTML files generated by Word/WordPerfect document converters. The purpose of the sample gawk script is to modify the HTML format of *General Supplemental Specification* files with the following changes to make them suitable for use in the ERL. The script itself is more suitable for creating new versions of HTML files from word documents. Simpler gawk scripts can be written to perform modifications to existing HTML files. Following functions need to be implemented in the Gawk script:

a. Delete or modify a number of unwanted format tags in the original HTML files to make the file look better or more suitable for further modifications by *gawk*. For the General Specs documents, such changes include:
   - Delete unwanted spaces between tags, such as the spaces between "<P>" and "<Center>" in "<P> <CENTER>".
   - Delete '<BR WP="BR1"><BR WP="BR2">', which creates unwanted blank lines within the htm file.
   - Delete redundant "<FONT>" tags. Two many "font" tags are used in the htm file. This makes the file lengthy and difficult to control.
b. Divide each of the original files, which contain one division of *gs*, in to smaller files each containing a section of the original file.
c. Create the HEAD part for each section file, including automatically generate the Title for each section.
d. Insert bookmarks to section titles and subtitles.
e. Insert links to Specifications, Materials I.M.'s, Articles, etc.
f. Add endings to each section file.

To implement the above functions, four script files were created, namely, *"preformat.awk"*, *"head.awk"*, *"body.awk"*, and *"end.awk"*. These files are to be run sequentially to generate the intended output files.

"Preformat.awk" performs the function of deleting unwanted formats in the original file.

"Head.awk" read in the original file as input and identify the section name for the different sections within the division file. From the section name, the output section file names are created and also the title for each of the section files. The head parts are written into each section file with a proper title.

"Body.awk" performs the functions described in d. and e. above.

"End.awk" generates the ending, like "</FONT>" and "</BODY>" for each of the section files.

One of the key points in developing *gawk* scripts for automatic processing HTML files is pattern definition. The computer must first identify certain patterns that appear in the texts to know what changes to be made to the html file. Sample scripts are listed in the index part of this manual.

D. Creating PDF Files

Converting Files to PDF Format

Some word files contain very complicated tables or are more oriented for printing. It may be a better idea to convert them into .pdf files than .htm files. Conversion to .pdf format is accomplished using Adobe Acrobat 4.0 or later. After the installation a new button called "Create Adobe PDF" will appear on the tool bar of the MS-Word application platform. This button will activate the **Acrobat PDFMaker 4.0 for Microsoft Word Dialogue Box** (Fig.

12) and enables us to convert a word file into PDF format. Selecting certain options when converting the file can retain links in the word file.

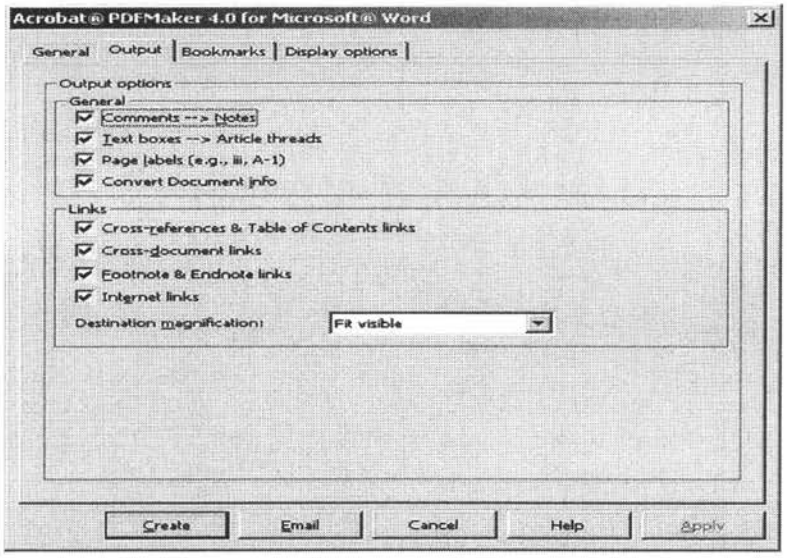


Fig. 12 Acrobat PDFMaker 4.0 for Microsoft Word Dialogue Box

4.4 Insert Links in PDF files

Links are inserted in a document using the link tool. The links can be specified as visible or invisible. To create a link:

1. Navigate to the section of the document where you want to create a link.
2. Select the link tool (Fig. 13). The pointer becomes a cross hair (+), and any existing links in the document, including invisible links are temporarily visible.

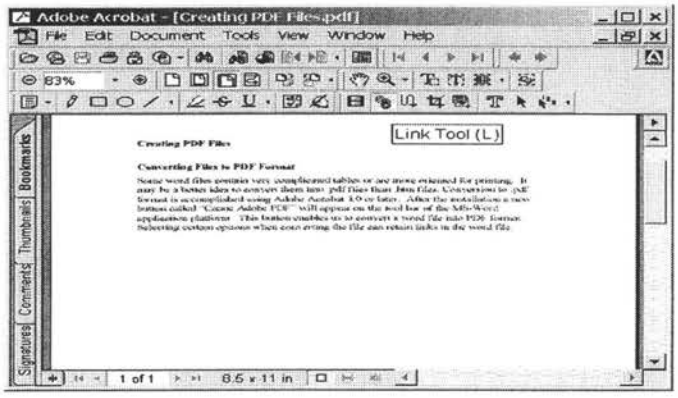


Fig. 13 Link Tool for creating links

3. Create the link rectangle in the following way:
4. Using the link tool draw a rectangle around the text where a link is to be added. Once the rectangle is drawn a Create Link dialog box appears (Fig. 14). In the **Create Link** dialog box, choose a rectangle type:

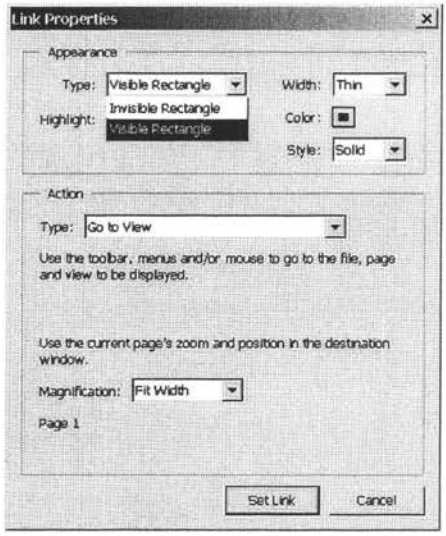


Fig. 14 Link Properties - Rectangle Type

5. **Visible Rectangle** (Fig. 15) indicates that the link rectangle is visible. Set the appearance of the link rectangle by choosing a width, color, and style. **Invisible Rectangle** indicates that the link rectangle should be invisible under normal circumstances.

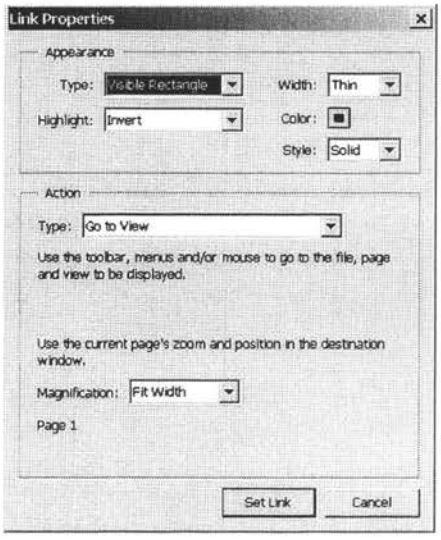


Fig. 15 Link Properties – Visible Rectangle

6. Select a highlight (Fig. 16) option for when the link is selected.

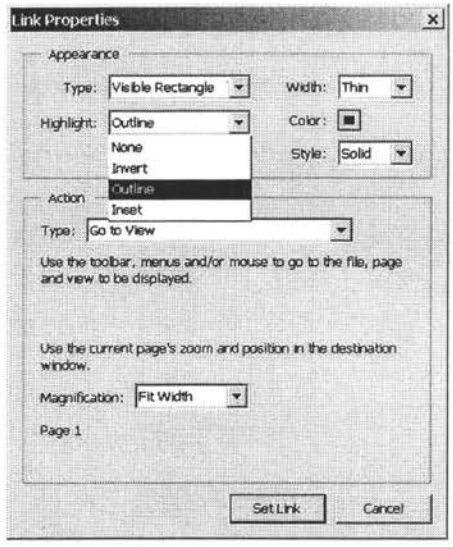


Fig. 16 Link Properties - Highlighting

7. Choose an action type (Fig. 17). This specifies the action that occurs when the link is selected. Choose **Open File**.

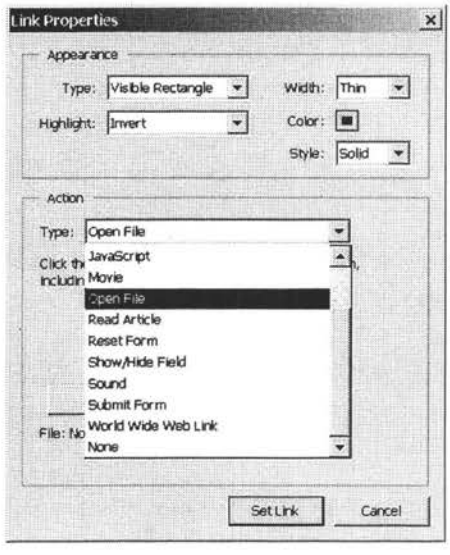


Fig. 17 Link Properties - Action Type

8. Click on select file. A **Select File to Open** dialogue box (Fig. 18) opens and select the file to be linked. Click **Set Link**.

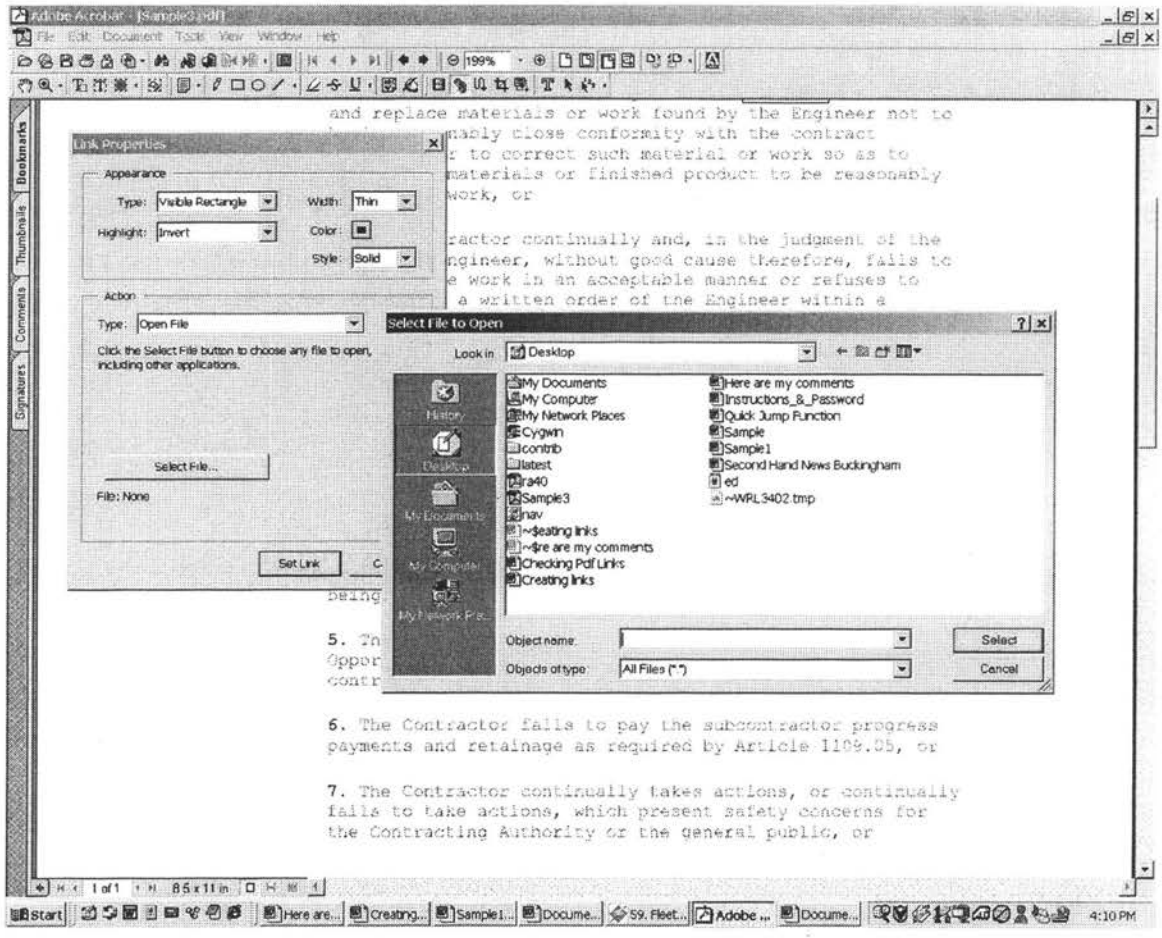


Fig. 18 Link Properties – Selecting file to open and set link

9. If you want to link your PDF document with another PDF document, use the **Go To View** action. Open the file in Acrobat and then navigate to the location where you want it to open.
10. Choose a magnification option. This allows you to control the view that appears when the link is selected.
11. Click **Set Link**.

4.5 Checking Pdf Links

The link checking needs to done for two sets of PDF files in the ERL.

1.IM

2.Road Plans

The link checking can be done by running a DOS batch file. The file is located in the Development directory. The file name is pdfLinkList and file type is

MS-DOS Batch File. Run the DOS batch file by double clicking it.

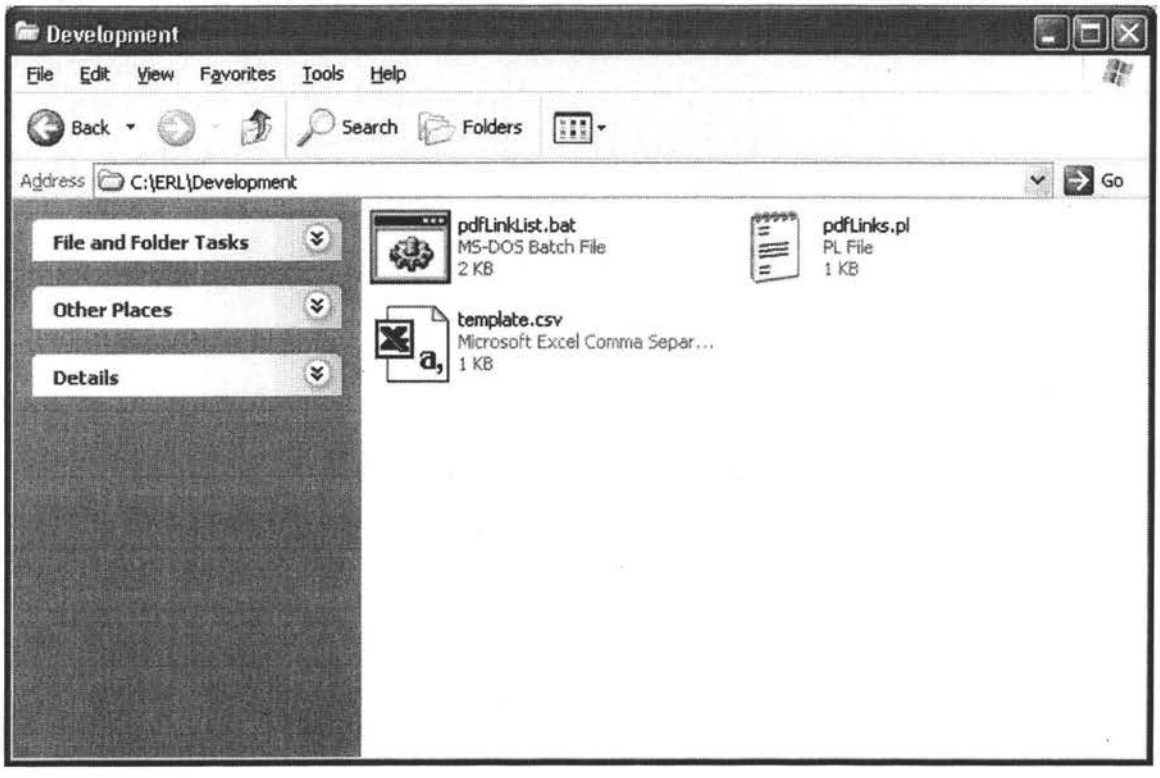


Fig. 19 MS – DOS Batch File location

The MS-DOS Batch File generates three files: Apr_2002_rs_eng.csv, Apr_2002_rs_met.csv, and Apr_2002_im.csv. All three files are readable using Microsoft Excel. Apr_2002_rs_eng.csv contains a list of all links for the English Road Standards, Apr_2002_rs_met.csv contains a list of links for the metric Road Standards, and Apr_2002_im.csv contains a list of links for the Material IMs. If Microsoft Excel is available, then double click on one of the three files to view its contents.

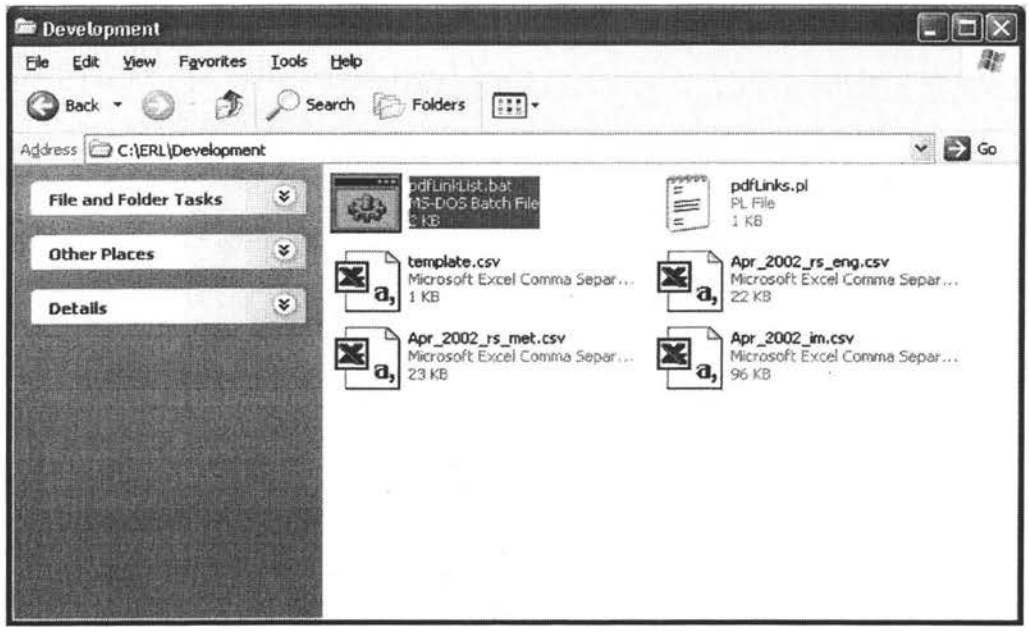
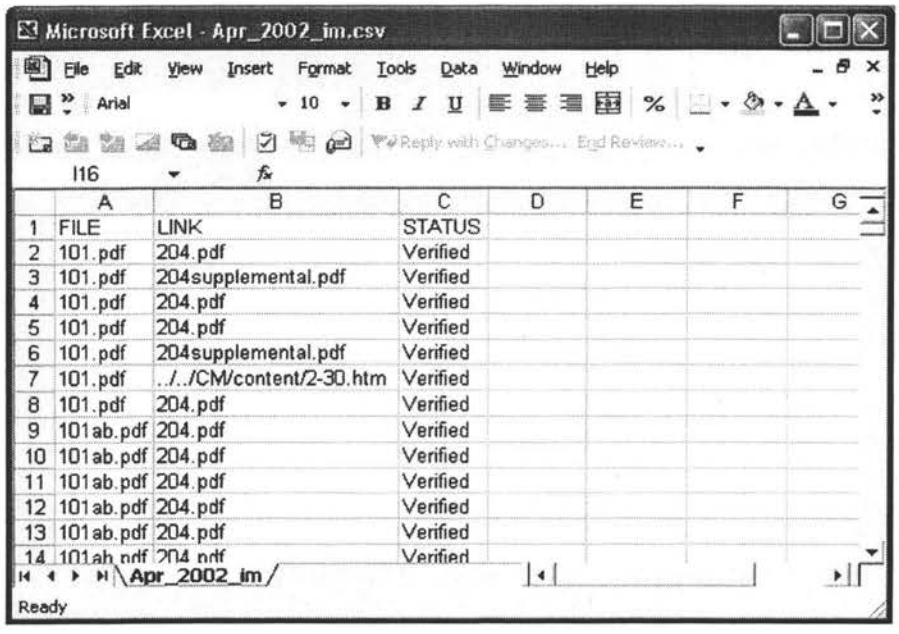


Fig. 20 csv file generation

The spreadsheet contains three columns. The first column lists the file names. If that file has any links, then those links are listed in the second column. The third column lists the status of the link. If the file exists, then that link will be listed as "Verified." If the file does not exist, then the link is listed as "Does not exist."



	A	B	C	D	E	F	G
1	FILE	LINK	STATUS				
2	101.pdf	204.pdf	Verified				
3	101.pdf	204supplemental.pdf	Verified				
4	101.pdf	204.pdf	Verified				
5	101.pdf	204.pdf	Verified				
6	101.pdf	204supplemental.pdf	Verified				
7	101.pdf	../../CM/content/2-30.htm	Verified				
8	101.pdf	204.pdf	Verified				
9	101ab.pdf	204.pdf	Verified				
10	101ab.pdf	204.pdf	Verified				
11	101ab.pdf	204.pdf	Verified				
12	101ab.pdf	204.pdf	Verified				
13	101ab.pdf	204.pdf	Verified				
14	101ab.pdf	204.pdf	Verified				

Fig. 21 List of PDF links

For this example, file 101.pdf from the Material IMs has seven links (Fig. 22). Four of the links are to file 204.pdf and two are from 204supplemental.pdf. Both of these links reference files in the same directory as file 101.pdf, which means the links are to other Material IM files from the same letting. The last link references file "../../cm/content/2-30.htm."One method of performing a quick check for link problems is to sort column 2 alphabetically. By sorting the list, the user can spot bad file and directory names more efficiently. If a suspicious link is found, then the link must be repaired using Adobe Acrobat. Please refer to the section on PDF file linking for further information.

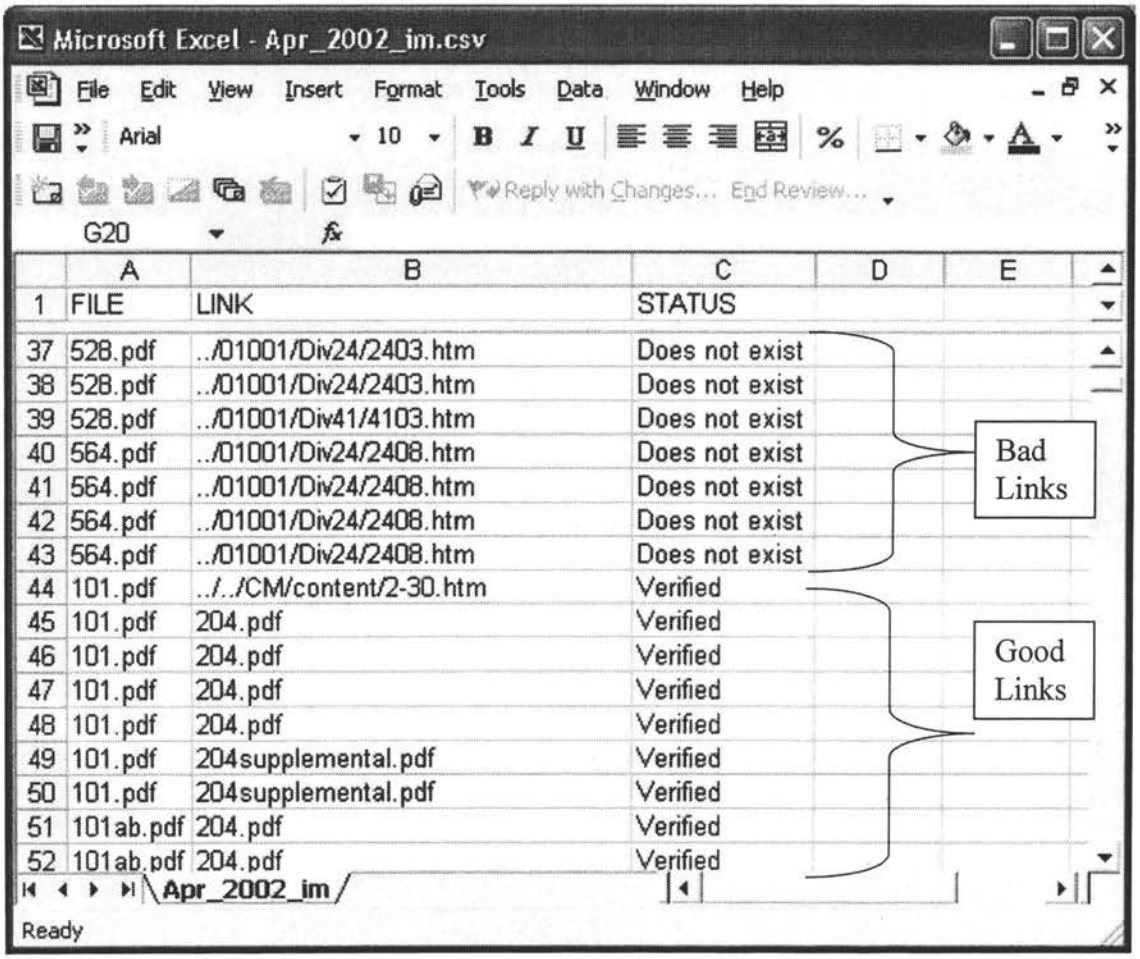


Fig. 22 Examples of good and bad links

Chapter 5. Working with Navigation Functions

The document files can be accessed via three ways, using the navigation bar, using the search engine, and using the Quick Jump function. Methods of implementing such functions will be discussed in this chapter.

5.1 Navigation Files

A navigation file is like the Table of Contents (TOC) in a book. It contains a list of section and subsection names to be covered in a document. These names are linked to the exact section files or subtitles of the documents. According to the complexity of the letting document, some documents have two levels of navigation files. The first level refers to the chapter or division names, through which the second level of navigation files are linked. And the second level navigation files contain section and subsection names of each chapter and division. By clicking these names, the reader can be directed to the exact place they want to find. The procedure of creating/updating navigation files goes like this:

1. Create/Update the text of the navigation files. Compare them with the TOC of the hardcopy and make sure that they match the document contents. Please note that during an update of a document, some chapters or sections may be added or deleted, and section names may be changed. Thus updater should compare the section names in the navigation file with those in the TOC in detail.
2. Add links to the navigation files. When doing this, make sure the URL uses relative path and the target files opens at the "main" frame of the frameset. To specify the target window of a link, put ' target = "frame name" ' in the <a href = "……"> tag.

   Example: <a href="../content/1100.htm" target="main">

5.2 Create and Deploy the Search Engine

The search engine is one of the most often used functions in the ERL, and it is also a major difference between an electronic version of documents and traditional books. A search engine package called *Quest Agent* is used in the ERL. A separate search engine is deployed

for each of the five documents included in the ERL. To learn how to install the *Quest Agent* software, read the **Appendix A: *Quest Agent 5.0 License Key & Installation Procedure***.

To create and deploy the search engine, follow the steps below:

1. Choose **Start – Programs – QuestAgent 5 – QuestManager**. The QuestManager interface will appear.
2. Select "**New Profile**" from the "**File**" menu to create new index profile. Wizard for index profile creation will appear. Follow instructions on the wizard.

For more information on how to use the wizard, deploy the search engine and customize the interface, choose **Start – Programs – QuestAgent 5 – index.htm** from windows desktop. Read the instructions on:

a. Getting started.
b. Deploying search applets
c. Customizing search applets.

After having generated and deployed the Search Engines, find out the file name to be used as the main search engine interface and make sure the right file name and path is referred in the navigation files (Fig. 23). The file name will be given in the deploying process. The default path for the file structure that is used in current ERL is: "../qamain2h.htm".

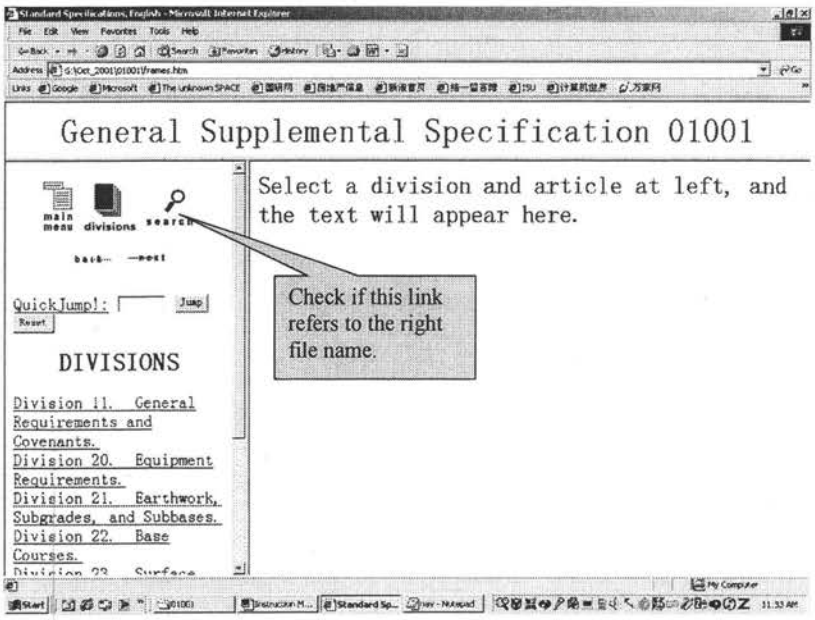


Fig. 23 Check the Search Icon in the Navigation Files

Two customizations are used in the user interface of the search engine.

a. Choose "[JS02H] Word Highlighting with Navigation" when asked which search interface template to use.

b. When the search engine has been created and deployed, change the following parameters in the "*qascr02.htm*" (under the document directory such as "gs", Fig. 24) file which will be generated by the *QuestAgent* using wordpad or notepad:

   i. Change the initial value of "showNavTags" from "false" to "true";

   ii. Change the initial value of "maxHits" from "100" to "25."

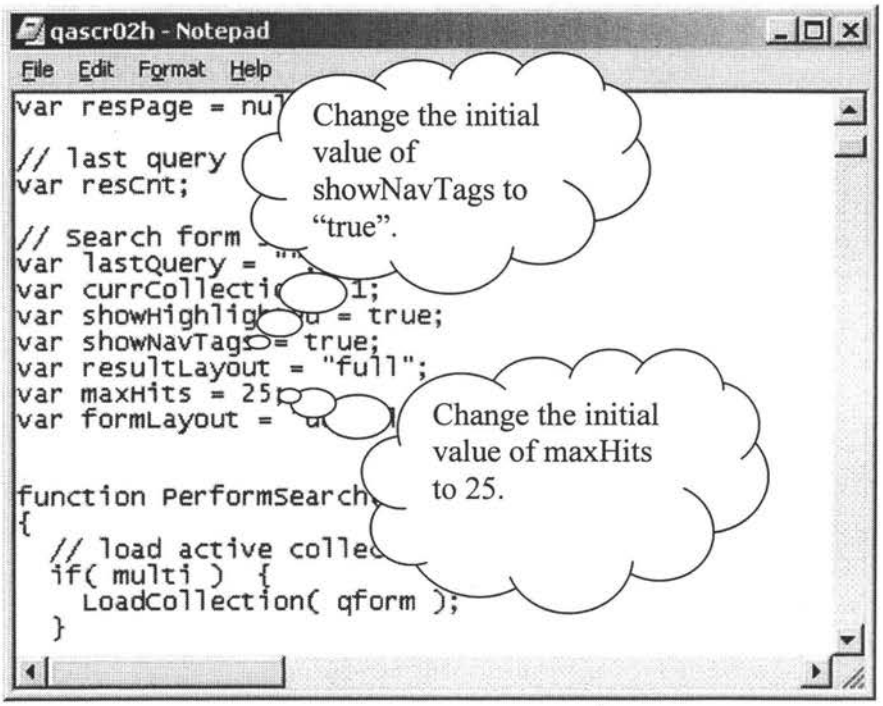


Fig. 24 Make Changes to qascr02.htm

These changes make the "Show navigation controls" check box be checked by default and modify the default maximum hit number from 100 to 25.

Find where the following codes appear in the same file:

```
doc.write( "        <option value=\"200\" " );
if( maxHits == 200 )
doc.write( " selected " );
```

```
doc.writeln( ">200</option>" );
doc.write( "        <option value=\"100\" " );
if( maxHits == 100 )
doc.write( " selected " );
doc.writeln( ">100</option>" );
doc.write( "        <option value=\"50\" " );
if( maxHits == 50 )
doc.write( " selected " );
doc.writeln( ">50</option>" );
doc.write( "        <option value=\"25\" " );
if( maxHits == 25 )
doc.write( " selected " );
doc.writeln( ">25</option>" );
```

Change the first three 200's to 25's, first three 100's to 50's, first three 50's to 100's, and first three 25's to 200's. This will change the order of the four maximum hit numbers in the search engine interface.

The final appearance of the search engine interface should look like Fig. 25.

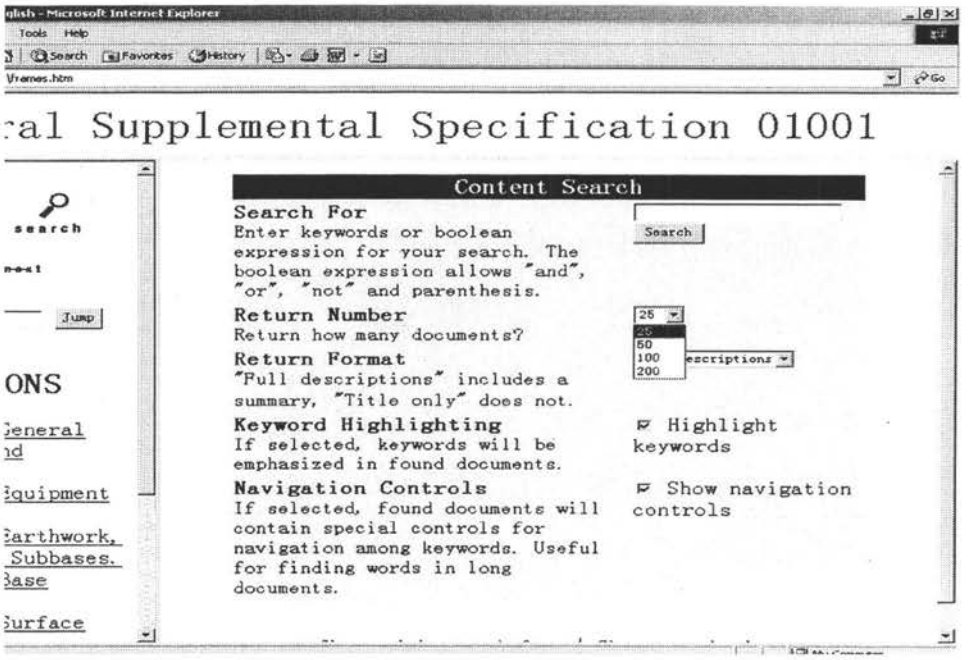


Fig. 25 The Search Engine Interface after Customization

5.3 Quick Jump Function

The following description of the quick jump function is valid for Standard Specification files only since this function is primarily used to navigate through the Standard Specification files. The quick jump feature is also enabled for the construction manual and the supplemental specifications files also.

The search can be done in 2 ways.

1. By entering a 4-digit format, for e.g.: 1101, where 1101 refers to Standard Specification section 1101.
2. By entering a 6-digit format, for e.g.: 1101.03, where 1101.03 refers to Standard Specification section 1101 subpart 03.

Four-Digit Format

When a four-digit number is entered in the quick jump, a list of the general requirements and covenants is generated. This is actually the navigation bar (Fig. 26) showing the section heading and subsections. Every time this format is used, the navigation bar is updated.

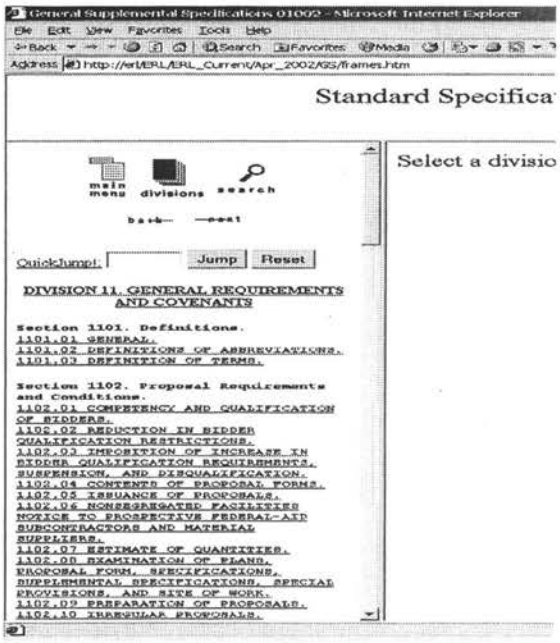


Fig. 26 Navigation bar showing section heading and subsections

Six-Digit Format

When a six-digit number in the format for e.g.: *1101.05* is entered in the quick jump, a list of the general requirements and covenants is generated in the main frame. This frame shows the subsections and the description of the each of them. Every time this format is used for search the main frame is updated.

Fig. 27 Main frame showing subsection description

When a character or a number is not entered in the above-mentioned format, for example *a23* is entered, an error message appears (**Error! Reference source not found.**) saying "OOPS!! You have entered an invalid section or article number. Please try again".

This error message also appears when a section number, which does not exist, is entered.

For example if *1101.05* is entered. This section does not exist and hence the error message is displayed.

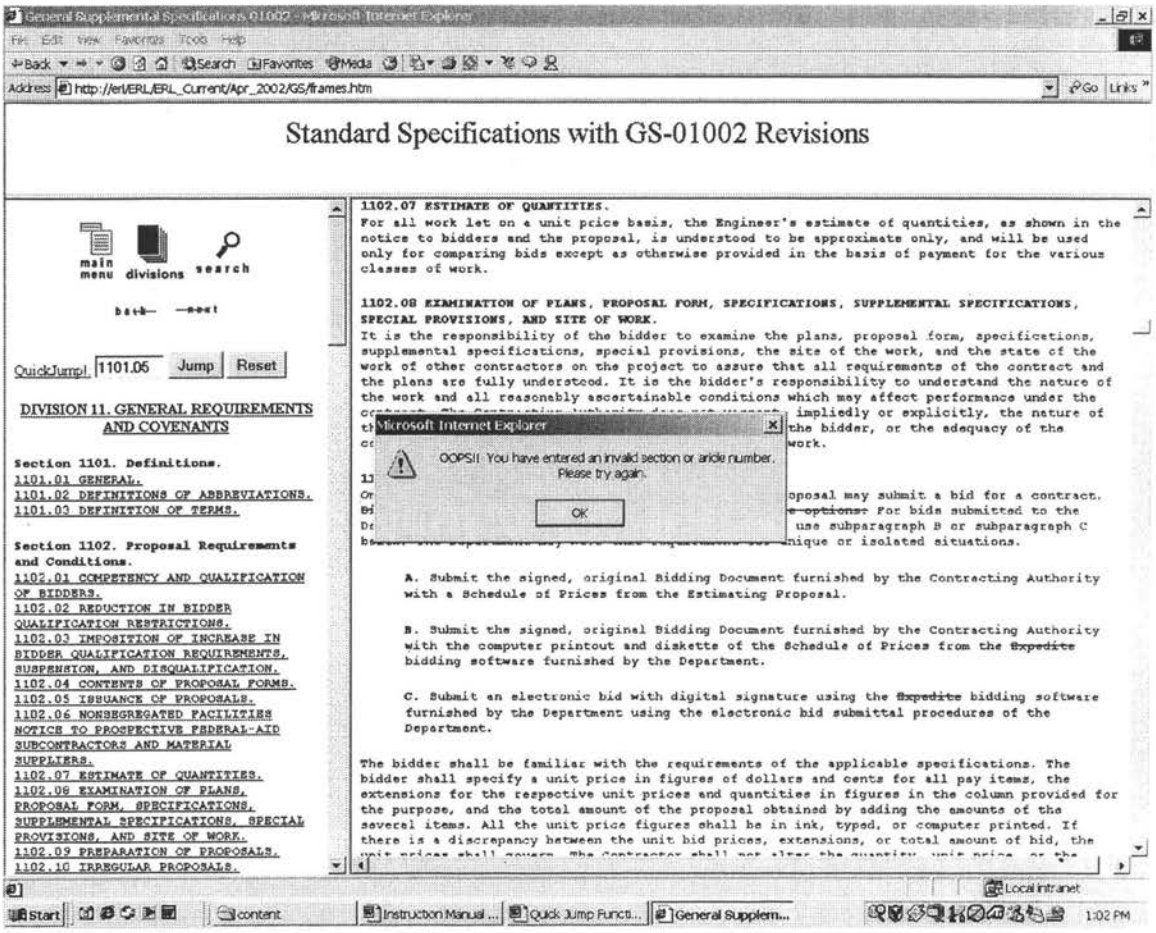


Fig. 28 Error message display

Function follow-up and maintenance

The quick jump function for the Standard specification file search is located at \gs\navigation and the file name is nav.js. The file is written using the JavaScript programming language. This file must be updated if any of the following events occur:

1. A specification section is removed
2. A specification section is added
3. Specification naming structure changes (such as changing the 4 digit format to a 5 digit format)
4. Filename format of the specification files are changed

Description of the QuickJump code

The QuickJump code is divided into two sections. The first section handles the 4 digit format. The code contains a line for each specification section. The following example shows the code for section 1108 through 1109.

```
{if (goHere == "1108") {window.location.replace("nav11.htm#1108");return;}};
{if (goHere == "1109") {window.location.replace("nav11.htm#1109");return;}};
{if (goHere == "1110") {window.location.replace("nav11.htm#1110");return;}};
{if (goHere == "1111") {window.location.replace("nav11.htm#1111");return;}};
```

The variable goHere contains the value that the ERL user entered into the quick jump form. Each line is executed successively until a match is found. If an ERL update adds new sections to the specifications, then the quick jump code must be modified to recognize those sections.

To add a new section for e.g. 1234, the code is as follows.

```
{if (goHere == "1234") {window.location.replace("nav12.htm#1234");return;}};
```

where the bold text indicates the differences from the previous examples.
This line of code is appended to the rest of the code as shown below.

```
{if (goHere == "1108") {window.location.replace("nav11.htm#1108");return;}};
{if (goHere == "1109") {window.location.replace("nav11.htm#1109");return;}};
{if (goHere == "1110") {window.location.replace("nav11.htm#1110");return;}};
{if (goHere == "1111") {window.location.replace("nav11.htm#1111");return;}};
{if (goHere == "1234") {window.location.replace("nav12.htm#1234");return;}};
```

The second part of the QuickJump code handles the 6 digit format.

```
{if (goHere == "1101.01") {{window.location.replace("nav11.htm#1101")};
{window.parent.frames["main"].location.replace("../content/1101.htm#1101.01")};return;}};

{if (goHere == "1101.02") {{window.location.replace("nav11.htm#1101")};
{window.parent.frames["main"].location.replace("../content/1101.htm#1101.02")};return;}};

{if (goHere == "1101.03") {{window.location.replace("nav11.htm#1101")};
{window.parent.frames["main"].location.replace("../content/1101.htm#1101.03")};return;}};
```

Again, the variable goHere contains the value that the ERL user entered into the QuickJump form, and each line is executed successively until a match is found. This code is slightly more complicated because two actions must occur: the navigation frame and the content frame are updated.

To add a new subsection for e.g. 1101.04, the code is as follows.

```
{if (goHere == "1234.56") {{window.location.replace("nav12.htm#1234")};
{window.parent.frames["main"].location.replace("../content/1234.htm#1234.56")};return;}};
```

where the changes are indicated with bold text.

This line of code is added to the rest of the code as shown below.

```
{if (goHere == "1101.01") {{window.location.replace("nav11.htm#1101")};
{window.parent.frames["main"].location.replace("../content/1101.htm#1101.01")};return;}};
{if (goHere == "1101.02") {{window.location.replace("nav11.htm#1101")};
{window.parent.frames["main"].location.replace("../content/1101.htm#1101.02")};return;}};
{if (goHere == "1101.03") {{window.location.replace("nav11.htm#1101")};
{window.parent.frames["main"].location.replace("../content/1101.htm#1101.03")};return;}};
{if (goHere == "1234.56") {{window.location.replace("nav12.htm#1234")};
{window.parent.frames["main"].location.replace("../content/1234.htm#1234.56")};return;}};
```

Chapter 6. Working with the Cover Page

The update of the opening page mainly consists of two parts: updating the image and updating the hierarchical menu.

6.1 Updating the Image

To update the image in the opening page, Adobe Photoshop 5.5 or above is recommended for editing the image. Edit operations include:

A. Cut the picture to 800×600 pixels.

To do this ():

a. Open the original image file in Adobe Photoshop. Click "Rectangular Marquee Tool" button at the Tool Bar.

b. Choose "Fixed Size" for "Style."

c. Type "800 px" for width and "600 px" for height.

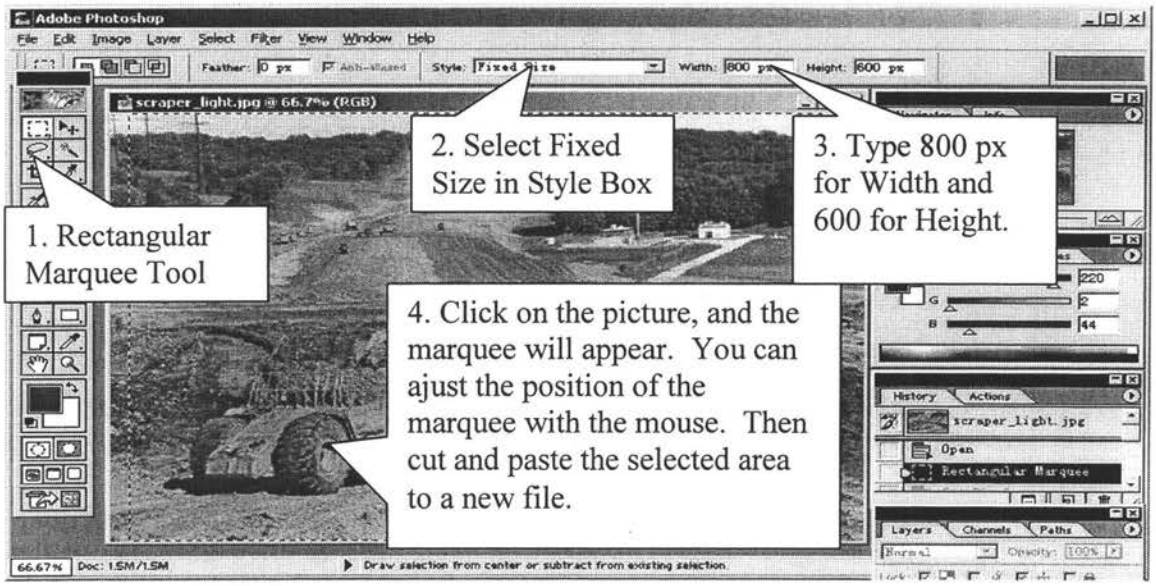


Fig. 29 Cut the Picture to 800×600 Pixels

d. Then click the picture with the mouse. An area surrounded by dashes will appear. Select Edit – Copy to copy the selected image to clipboard. Open a new file. By

default the new image file has a size of 800 by 600 pixels. Choose Edit – Paste to paste the selected image to the new file.

B. Add DOT Logo and Texts to the Image.

To do this, you either have a dot logo image available to you or you will have to create it from the previous pictures by yourself. An IDOT logo will be provided with this manual.

To cut the logo out:

a. Open the background image and the IDOT logo in Photoshop.
b. Set the color mode to be RGB by selecting "**Image – Mode – RGB color**".
c. Activate the IDOT logo layer. Press **Ctrl+A** or Choose "**Select – All**" to select all image in the layer. Copy and Paste the selected image to the background picture.
d. Click the Move Tool on the Tool Bar, and make the logo layer. Then move the logo to the proper position.

To add texts to the image file:

a. Click the "Type Tool" button on the tool bar. Draw a rectangle on the image, select the right color, font and size for the texts, and begin typing texts in the text box.
b. Click the "Move Tool" button on the tool bar to adjust the position of the text on the image. Save file as "Start_Here.jpg". Choose "OK" when asked if the image is to be flattened.

Fig. 30 Add DOT Logo to the Picture

c. Slice the Image and Edit the Images for the "Read Me," "User's Guide," "Phone Books," and "Credits."

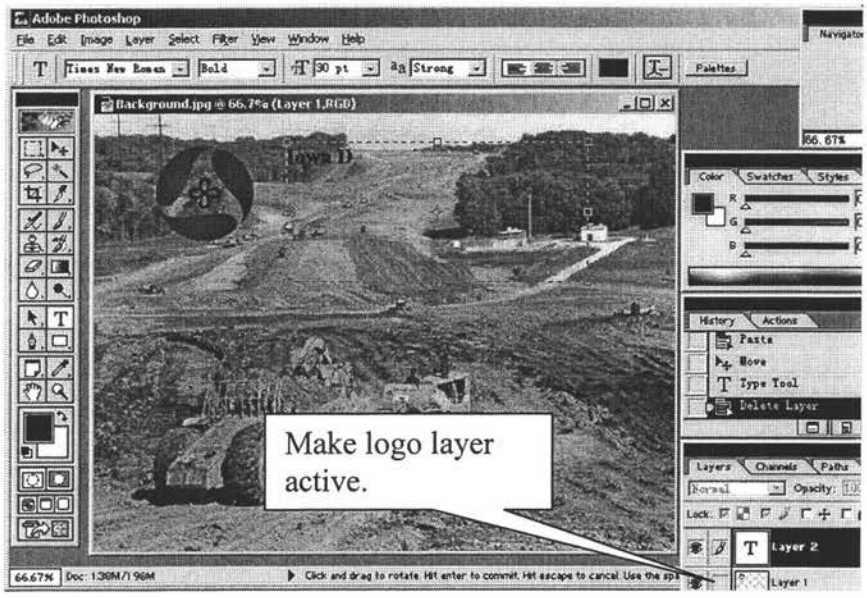


Fig. 31 Add Texts to the Picture

To slice the image:

1. Right click the "Slice Tool" button. Choose "Slice Tool" (Fig. 32). Right-click the piece of image to be sliced.
2. Select "Slice Options." In the Dimensions block, type in the widths and heights for the slice dimension (Fig. 33). Continue this process until the photo is sliced to pieces of desired sizes. For dimensions of each slice, please refer to the image size of earlier versions of the opening page. Current image size used in the opening pages is:

1	800 × 300 px	6	125 × 26 px
2	575 × 300 px	7	125 × 26 px
3	125 × 180 px	8	125 × 26 px
4	100 × 300 px	9	125 × 16 px
5	125 × 26 px		

3. Then click File – Save for Web, and save the image as "Start_Here." A web file named Start_Here.htm will be generated in this process.

Fig. 32 Cut the Picture into Slices

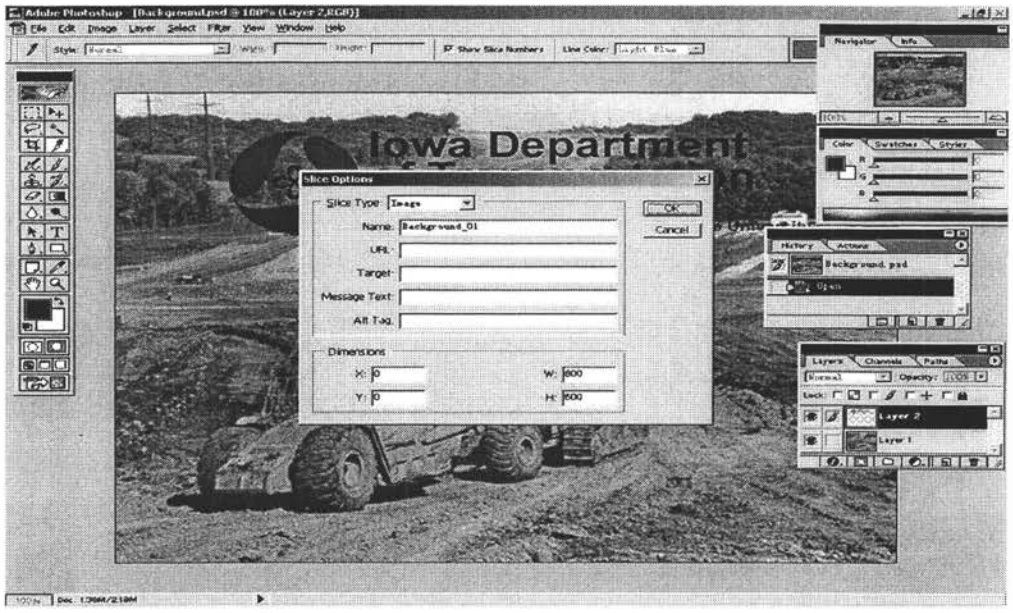


Fig. 33 Enter the Size for Each Slice of the Picture

We will use the Read Me case (Fig. 34) to illustrate how to add texts to image slices for the Read Me, User's Guide, Phone Book, Credits:

1. Open the image slice on which "Read Me" should be, slice 5 in this case, in Photoshop. Add text "Read Me" in on the image and adjust its position. Make sure the texts are in red. Save the image as "read.gif" and "read3.gif."
2. Then open the original image (with no text) again and add "Read Me" in the desired color on the image and position the text a little (about 6 pixels) left to the position of the red "Read Me." Save the text in the name of "read2.gif."

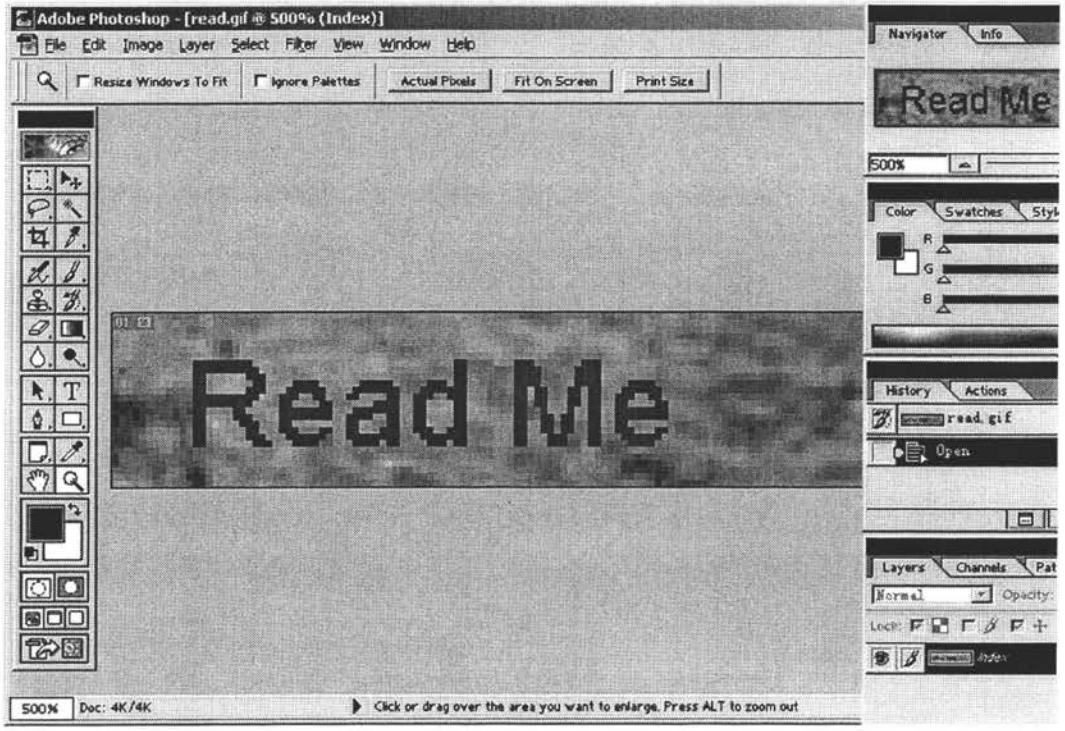


Fig. 34 Adding Texts to the Read Me Slice

3. Repeat this process for the "User's Guide," "Phone Book," "Credits." When saving the image files, use the exact file name used for their counterparts in earlier versions, because these file names will be referred to in the "Start_Here.htm" file. The file names currently used for the above image slices are: guide[2/3].gif, phone[2/3].gif, and credit[2/3].gif.

6.2 The Hierarchical Menu

A. Introduction

When you place the mouse on the "Start Here" box on the opening page after the page is

fully loaded, a two level menu appears which allows you to select a document name to go to. This menu is called a "Hier Menu". The Hier Menu is a free JavaScript software tool that can be used on web pages. More information about Hier Menu can be found on: http://www.dhtmlab.com/. The version currently used on the current ERL (October 2001) is 4.0.12. You can also download the latest version of Hier Menu from http://www.dhtmlab.com/. The HierMenu had been free of charge on the condition that their names are mentioned as the owner of the HierMenu and the JavaScript codes are not altered. But now they have started charging money for use of the HierMenu.

After having downloaded and extracted the Hier Menu scripts and pictures to a directory, you can see there are one htm file, five JavaScript files and a number of arrow images to be used in the script files. To learn about how the Hier Menu can be included in the Opening Page of the ERL, we should first have a careful study of the example htm page (LoadMe.html).

Double click the LoadMe.html file. The file will be opened by the default web browser. There are five types of Hier Menus used in the example page. What is used in the ERL is the one called "Absolute Position Popup Menu (Displays on Mouseover)."

Click "View-Source" on the Browser menu (see Fig. 36.), the source file of the LoadMe.html will be opened. The majority of the source file scripts should be included in the Cover Page of the ERL, except a few lines showing other types of Hier Menu formats. Carefully read the scripts and identify the function of each part of the script. Three parts of codes are connected to the type of menu that we use. The codes and function of these parts are shown below.

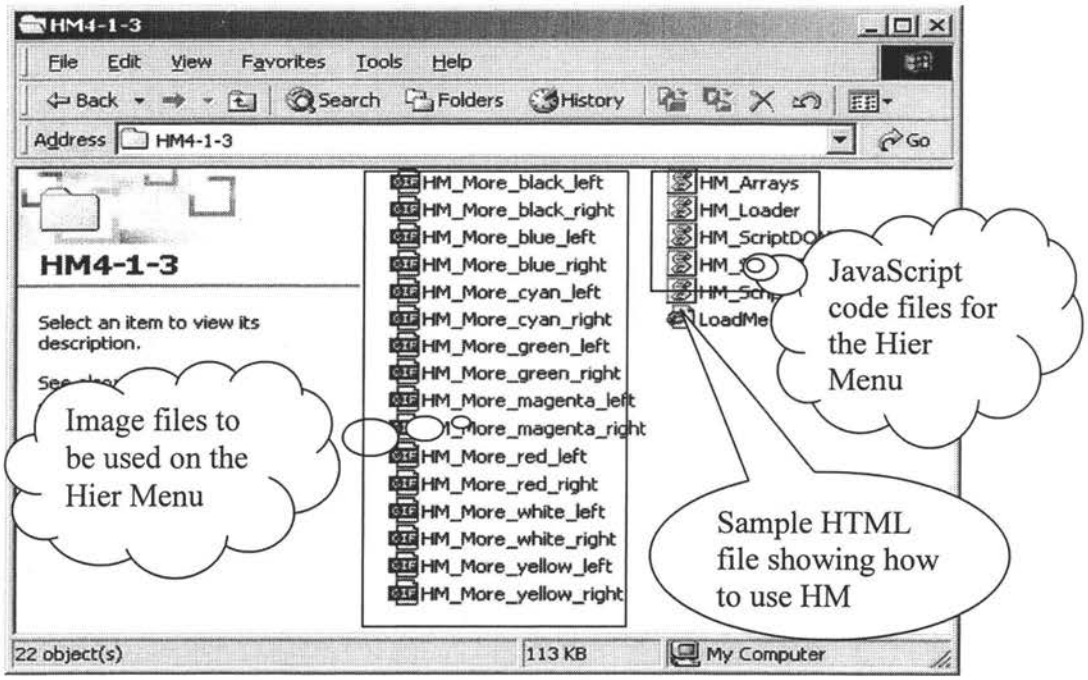


Fig. 35 Files Extracted for HierMenu

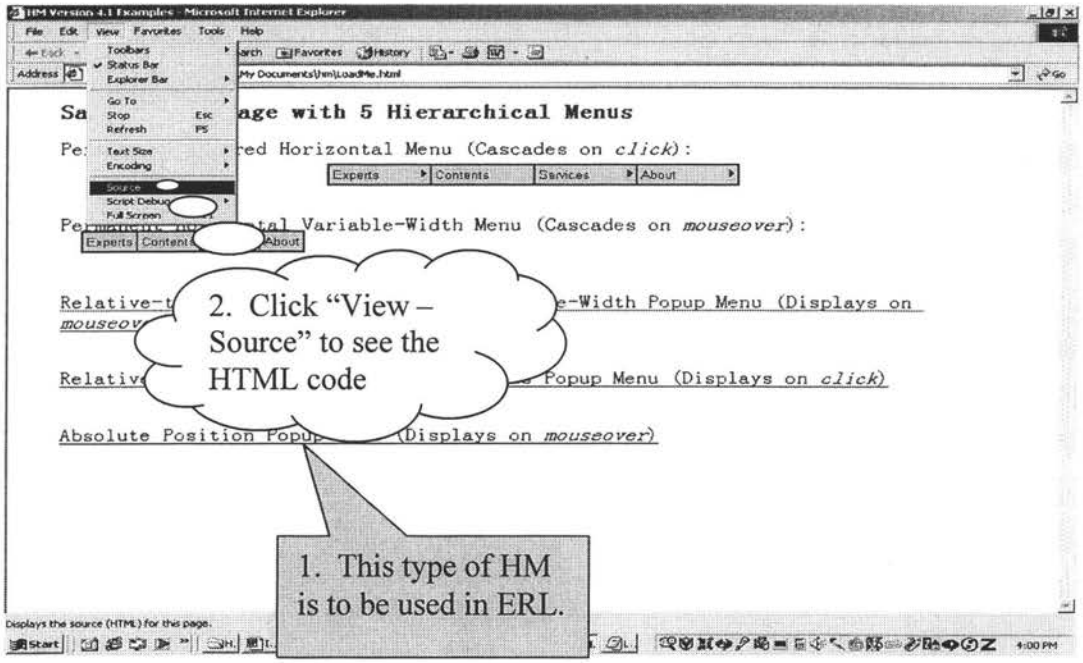


Fig. 36 Sample HTML of Using Hierarchical Menus *LoadMe.html*

Part 1:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
if(window.event + "" == "undefined") event = null;
function HM_f_PopUp(){return false};
function HM_f_PopDown(){return false};
popUp = HM_f_PopUp;
popDown = HM_f_PopDown;
//-->
</SCRIPT>
<SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
<!--
HM_PG_MenuWidth = 150;
HM_PG_FontFamily = "Arial,sans-serif";
HM_PG_FontSize = 10;
HM_PG_FontBold = 0;
HM_PG_FontItalic = 0;
HM_PG_FontColor = "blue";
HM_PG_FontColorOver = "white";
HM_PG_BGColor = "#DDDDDD";
HM_PG_BGColorOver = "#FFCCCC";
HM_PG_ItemPadding = 3;

HM_PG_BorderWidth = 2;
HM_PG_BorderColor = "black";
HM_PG_BorderStyle = "solid";
HM_PG_SeparatorSize = 2;
HM_PG_SeparatorColor = "#d0ff00";
HM_PG_ImageSrc = "Coverpage/HM4/tri.gif";
HM_PG_ImageSrcLeft = "Coverpage/HM4/triL.gif";

HM_PG_ImageSize = 10;
HM_PG_ImageHorizSpace = 0;
HM_PG_ImageVertSpace = 2;

HM_PG_KeepHilite = true;
```

```
HM_PG_ClickStart = 0;
HM_PG_ClickKill = true;
HM_PG_ChildOverlap = 3;
HM_PG_ChildOffset = 0;
HM_PG_ChildPerCentOver = null;
HM_PG_TopSecondsVisible = .5;
HM_PG_StatusDisplayBuild =0;
HM_PG_StatusDisplayLink = 0;
HM_PG_UponDisplay = null;
HM_PG_UponHide = null;
HM_PG_RightToLeft = false;
//HM_PG_CreateTopOnly = 1;
HM_PG_ShowLinkCursor = 1;
//HM_a_TreesToBuild = [1,2];
//-->
</SCRIPT>
```

Part 2:

```
<area href="Start_Here.htm" coords="275, 266, 477, 299" shape="rect" alt="[Menu]"
onMouseOver="popUp('HM_Menu3',event)" onMouseOut="popDown('HM_Menu3')"
onClick="return false">
```

Part 3:

```
<SCRIPT LANGUAGE="JavaScript1.2" SRC="Coverpage/HM4/HM_Loader.js"
TYPE='text/javascript'></SCRIPT>
```

B. Add Hierarchical Menu to the Opening Page

Copy all the files downloaded except LoadMe.html to "*Coverpage\HM4*". Now we can include the Hier Menu in the Opening Page by modifying the HTML codes in earlier version of the Opening Page.

In most cases, it would be a good idea to start your work on the bases of the opening page of previous versions of ERL. The three parts of scripts described in the previous section should be copied and pasted to the proper places in the Opening Page HTML scripts. To find where these scripts should be placed, you can refer to the HTML code of current ERL (Fig. 37).

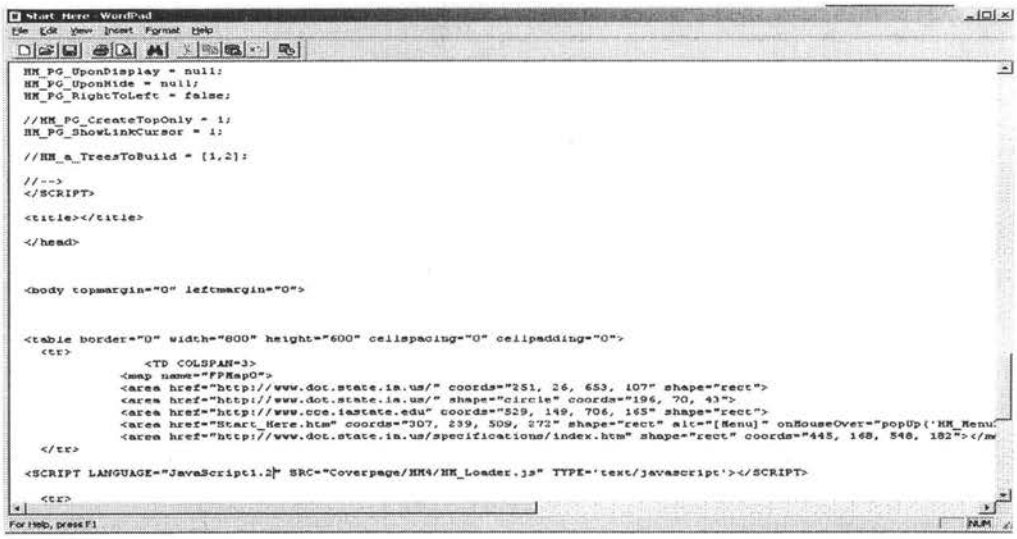


Fig. 37 Insert Hier Menu Scripts to the Proper Position of the Cover Page

The file HM_Arrays.js defines the texts, links and format of the Hierarchical Menu. To update the contents and appearance of the Hier Menu, open the file named "*HM_Arrays.js*" in the directory "*Coverpage\HM4*" using notepad (see Fig. 38.). Edit the texts to incorporate new changes in the ERL.

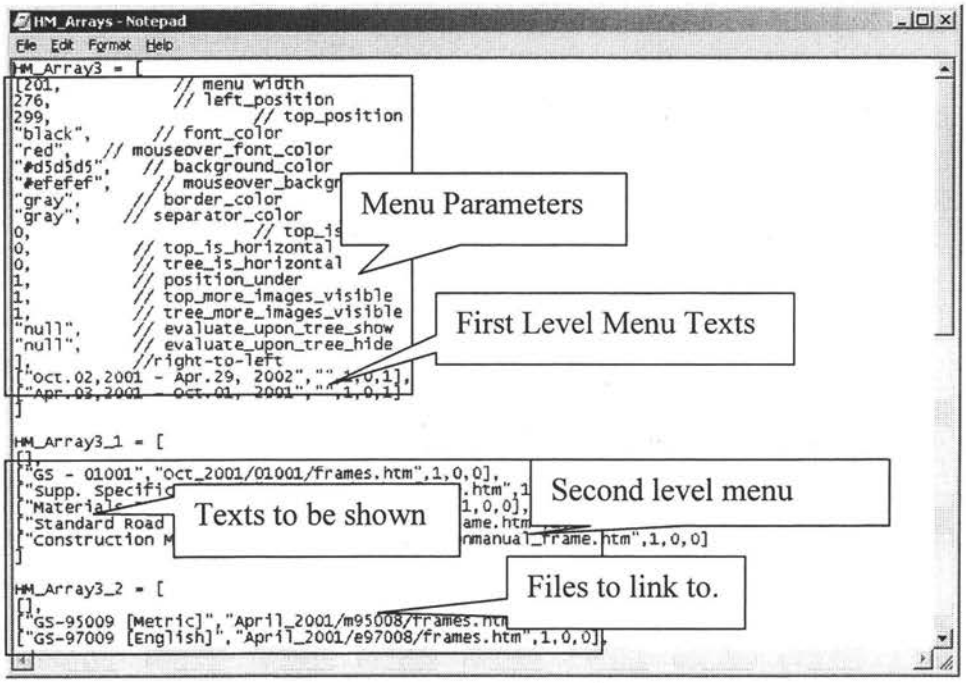


Fig. 38 Modifying HM_Arrays.js

6.3 Insert Map Areas and Hyperlinks to Iowa DOT and ISU web sites

Besides including the Hier Menu codes, we also need to add some links on the web page to Iowa DOT and ISU web sites. To do this (see Fig. 39.):

1. Open Start_Here.htm in FrontPage.
2. Click the right type of hotspot (rectangle or circle), which is on the "pictures" toolbar, which you would like to insert on the image. If the "pictures" toolbar is not shown on the screen, activate the toolbar by clicking "View – Toolbars – Pictures".
3. Then draw a circle or rectangle around the texts or logo where you would like to insert a link. After the shape is drawn, an "Edit Hyperlink" dialogue box will jump out (see Fig. 40.). Put in the URL you wish to link.

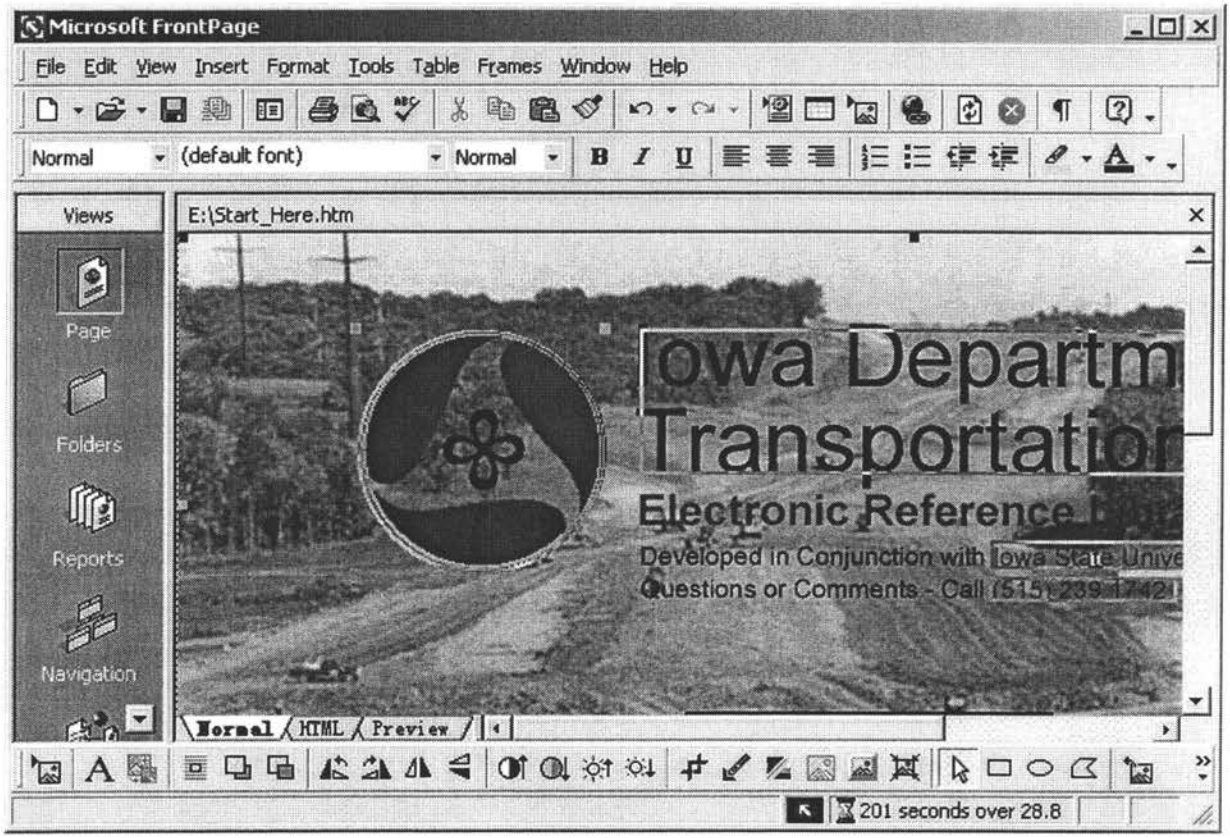


Fig. 39 Add Links on the Opening Page

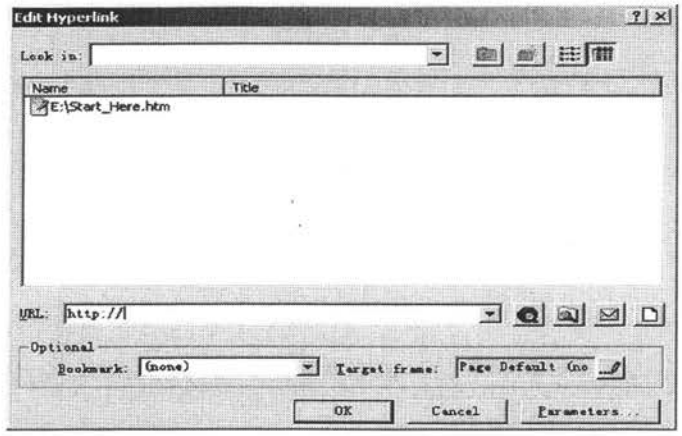


Fig. 40 Edit Hyperlink Dialogue Box

Now the Cover Page is complete.

Chapter 7. Error Checking

There are around 5,000 files and 21,000 links in an ERL. To find mistakes in so many files is really a burdensome work. A systematic method of checking and correcting errors must be developed.

A major way of eliminating errors is to generate fewer errors in the updating process. To achieve this, the following points should be noted:

1. Careful prior planning and standardization are helpful principles for error elimination.
2. Properly control the features of input files, including format used, file name, etc.
3. Determine the file structure using the suggested template before the updating operation begins. Avoid changing directory names or file structure.
4. Always keep in mind to use relative paths [4]instead of absolute paths[5].
5. Tightly control the update progress. Finish the updating process as early as possible to allow more time for error checking.

Suggested methods of checking errors after the updating has been completed include:

1. Using the summary report function provided by Microsoft FrontPage. Two statistics are the major sources of finding errors, "Unlinked Files" and "Broken Hyperlinks". By double clicking each of the two rows of data, you can get a detailed report on unlinked files or broken hyperlinks. Not all listed on the reports are mistakes. You can check and find out why they are listed and correct the mistakes.

---

[4] Relative Path: Use the file path from the anchor file to locate the target file. Suppose "navigation" and "content" are two directories under "gs"; "1103.htm" is a file in "content". To link to "1103.htm" from an anchor in a file in "navigation", the relative path is "../content/1103.htm".

[5] Absolute Path: Use the file path with the absolute name of the hard drives as a basic reference, i.e. "C:\ERL_2002\Apr_2002\gs\content\1103.htm".

2. Links in PDF files cannot be reported in FrontPage. For these links, we can generate a list of links by using scripts. Then we can find out if all links (paths) are correctly written by examining the list instead of going through the PDF files one by one.
3. Cross checking, that is, if more than one person updates the documents, the updater needs to check the work of the others and log checking result.
4. Manually compare navigation files with the Table of Contents of a hardcopy of a document. An express list of changes to the document will be of great help for both updating and checking errors. Click on each hyperlink on the navigation bar. Check if they link to the right files and these files are shown in the right frame.
5. Manually check some individual functions such as the search engine, Quick Jump, and Opening Page.
6. Check the opening page. See if images appears correctly, all necessary texts are included, the hyperlinks link to the right web sites or pages within the CD, and that the hier menu is correctly updated.
7. Check the quick jump function in the navigation files, make sure it can jump to sections newly added to the documents. Also be sure the error message appears correctly when an invalid entry is made.
8. Check the search engine. Check the interface and test if its search function works properly.

Up to now, the major error types encountered by developers include:

1. Use absolute path instead of relative path names.
2. Links are not updated to match file structure change, like changes of directory or file names
3. Navigation files do not match the Table of Contents.
4. Files are infected with virus.
5. Letter cases in hyperlinks do not match that of directory and file names.

When checking errors, we can intentionally check whether such types of errors exist and eliminate them.

Chapter 8. Suggestions on Future Improvement

The following aspects may be good sources for future improvements for the ERL:

1. Larger storage media, like DVD ROM.
2. An all-year round survey form or feedback communication channel to gather user suggestions and information.
3. Developing a usability test to find ideas for future improvement.
4. Further automation of the Update process to improve efficiency. Up to date, tools that have been found useful for automating the update process include, gawk, Perl.

Appendix A: Quest Agent 5.0 License Key & Installation Procedure

1. Download and install the latest revision of QuestAgent 5.0. (That is the same software installer that you used to evaluate the software.)

2. Extract ZIP file attached to this email to the <QuestAgent>/config directory, where <QuestAgent> is QuestAgent 5.0 installation directory. Default installation directory on Windows platform is "C:\Program Files\JObjects\QuestAgent50".

3. Start QuestManager application, select "Enter Serial" option from the "Help" menu and enter your serial code: 92PM3YTE

4. Restart QuestManager in order for changes to take effect.

5. Verify that license key is correctly installed. Choose "About" option from the "Help" menu and check shown license details. If software license is still marked as "evaluation", repeat the key installation procedure or contact support@jobjects.com for assistance.

==========

QuestAgent Home: ----------
http://www.jobjects.com/products/questagent/index.html

QuestAgent Support: ----------
Please use link bellow for FAQs and other support issues: http://www.jobjects.com/support/questagent/index.html

Download Updates: ----------

Update for the latest improvements, bug fixes and new features can be downloaded from:

http://www.jobjects.com/downloads/qa50/updates.html

Appendix B: Gawk Scripts

Preformat:

```
#This script deletes unwanted formats in the HTML file.
{
gsub(">[ ]*<", "><", $0);
gsub("<BODY[ A-Za-z0-9=#\"]*>", "<BODY>", $0);
gsub("<BR WP=\"BR1\"><BR WP=\"BR2\">", "", $0);
gsub("<FONT [ A-Za-z0-9=#\"\-]*>", "", $0);
gsub("<\/FONT>", "",$0);
gsub(" ", " ", $0);
print $0 > "./prefmt/"FILENAME;
}
```

Head:

```
# generate title

{if (FNR == 1) {Title="";Filename="";}}
{if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/DIVISION/) {
  Divn=substr($2,1,2);
  SN=Divn"00";
  FileName="./gs/"SN".htm";
  sub(/<P>/, "", $0);
  sub(/<CENTER>/, "", $0);
  sub(/<STRONG>/, "", $0);
  sub(/<\/STRONG><\/CENTER>/, "", $0);
  Title=$0;
  print "<HTML>" > FileName;
  print "<HEAD>" > FileName;
  print "<META HTTP-EQUIV=\"Content-Type\" CONTENT=\"text\/html; charset=windows-
1252\">" > FileName;
  print "<META NAME=\"Generator\" CONTENT=\"Corel WordPerfect 8\">" > FileName;
  print "<TITLE>"Title"<\/TITLE>" > FileName;
  print "</HEAD>" > FileName;
  print "<BODY>" > FileName;
```

```
    print "<Font FACE=\"Courier New\" SIZE=\"2\">" > FileName;
            }
}
{if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/Section/) {
  {
        for (i=1; i<=NF; i++) {
            if ($i~/Section/) SN=substr($(i+1),1,4);
                    }
  }
  FileName="./gs/"SN".htm";
  sub(/<P><CENTER><STRONG>/, "", $0);
  sub(/<\/STRONG><\/CENTER>/, "", $0);
  Title=$0;
  print "<HTML>" > FileName;
  print "<HEAD>" > FileName;
  print "<META HTTP-EQUIV=\"Content-Type\" CONTENT=\"text\/html; charset=windows-1252\">"
> FileName;
  print "<META NAME=\"Generator\" CONTENT=\"Corel WordPerfect 8\">" > FileName;
  print "<TITLE>"Title"<\/TITLE>" > FileName;
  print "</HEAD>" > FileName;
  print "<BODY>" > FileName;
  print "<Font FACE=\"Courier New\" SIZE=\"2\">" > FileName;
        }
}
```

Body:

```
#Identify the correct file name.
{if (FNR == 1) {Title=""; Filename=""; SN=""; ssn="";}}
{if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/DIVISION/) {
  {
        for (i=1; i<=NF; i++) {
            if ($i~/DIVISION/)      {
                    Divn=substr($(i+1),1,2);
                    SN=Divn"00";
                        }
                    }
```

```
  }
  FileName="./gs/"SN".htm";
  print FileName;}
}
{if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/Section/) {
  {
        for (i=1; i<=NF; i++) {
              if ($i~/Section/) SN=substr($(i+1),1,4);
                                    }
  }
  FileName="./gs/"SN".htm";
  print FileName;}
}

#Bookmark a Section Title
/^<P><STRONG>[0-9][0-9][0-9][0-9].[0-9][0-9][ ]*[A-Z]*/     {
  split($1, a, "<STRONG>");
  ssn=a[2];
  name=ssn;
  sub(/<STRONG>/, "<STRONG><a name=\""name"\">", $0);
  sub(/<\/STRONG>/,"<\/a><\/STRONG>", $0);
  }

#Bookmark a SubTitle

  {
  if ($0~/<P>[ \t]*<STRONG>[ .\t]*[A-Z]\.[ \t]/)  {print $0;
        for (i=1; i<=NF; i++)      {if      ($i~/[A-Z]\./)      {subttl=substr($i,(length($i)-
1),1);name=ssn""subttl;print subttl;print name;}}
        sub(/<STRONG>/, "<STRONG><a name=\""name"\">", $0);
        sub(/<\/STRONG>/,"<\/a><\/STRONG>", $0);
                                    }
  }
```

```
# Make links for strings with "S/specification", "Materials I.M.", "Article"
{for (i=1;i<= NF; i++)        {
  if ($i ~ /Article[s]*/ && $(i+1) ~ /[0-9][0-9][0-9][1-9][ ,\.][^0-9]/)
            {
            if (length($(i+1))<=6) {
                  div1=substr($(i+j),1,2);
                  div2=substr($(i+j),1,4);
                  sub(div1, "<a href=\"..\/Div"div1"\/"div2".htm\">"div1, $(i+j));
                  $(i+j)=$(i+j)"<\/a>";
                        }

            else {
                  if ($(i+1)~/[A-Z][,\.]/

{for (j=1; j<=(NF-i); j++)
                  {       if ($(i+j)~/[0-9][0-9][0-9][1-9][.]*/)
                        {
                              div1=substr($(i+j),1,2);
                              div2=substr($(i+j),1,4);
                              div3=substr($(i+j),1,7);
                              sub(div1,                                    "<a
href=\"..\/Div"div1"\/"div2".htm#"substr($(i+j),1,7)"\">"div1, $(i+j)); $(i+j)=$(i+j)"<\/a>"}
                              }
                        }
            }

# if ($i ~ /Article[s]*/ && $(i+1) ~ /[0-9][0-9][0-9][1-9][.]*/)
#           {
#                 div1=substr($(i+1),1,2);
#                 div2=substr($(i+1),1,4);
#                 bkmk=substr($(i+1),1,7);
```

```
        #                       if ($(i+2)
        #                       sub(div1, "<a href=\"..\/Div"div1"\/"div2".htm#"substr($(i+1),1,7)"\">"div1,
$(i+1)); $(i+1)=$(i+1)"<\/a>"}
        # }
        }

        # Make links to Materials I.M.
        {for (i=1;i<= NF; i++)          {
          if ($i ~ /[Mm]aterials/ && $(i+1) ~/I.M./)
                  {
                  split($(i+2), ima, "<");
                  {if (length(ima[1])<=4) {im=substr(ima[1], 1, 3);}}
                  {if (length(ima[1])>=6) {im=substr(ima[1], 1, 6);}}
                  {if          ($(i+3)~/Appendix/)          {sub(/[Mm]aterials/,          "<a
href=\"..\/..\/Im\/"im"a"substr($(i+4),1,1)".pdf\">Materials", $i);$(i+4)=$(i+4)"<\/a>"}
                  else        {sub(/[Mm]aterials/,        "<a        href=\"..\/..\/Im\/"im".pdf\">Materials",
$i);$(i+2)=$(i+2)"<\/a>";}}
                  }
                        }
        }

        {if (FileName!="") {print $0 >>FileName;}}


        End:
        # generate ending

        {if (FNR == 1) {Title="";Filename="";}}
        {if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/DIVISION/) {
          Divn=substr($2,1,2);
          SN=Divn"00";
          FileName="./gs/"SN".htm";
          print "</FONT>" >> FileName;
          print "</BODY>" >> FileName;
                  }
        }
```

```
{if ($0 ~/<P>/ && $0~/<CENTER>/ && $0~/<STRONG>/ && $0~/Section/) {
  SN=substr($2,1,4);
  FileName="./gs/"SN".htm";
  print "</FONT>" >> FileName;
  print "</BODY>" >> FileName;
          }
}
```

References

Li, Chun 2000. Developing a Hypertext-Based Electronic Reference Library for a Public Transportation Agency, Thesis for M.S. degree, Iowa State University.

Cetin, Ozdemir 2001. Improvements in the Hypertext-based Electronic Reference Library for a Public Transportation Agency, Thesis for M.S. degree, Iowa State University.

Dynamic HTML Lab, 2002, DHTML Lab – Hiermenus Center, http://www.dhtmlab.com/, date accessed July 16, 2002.

GNU Project, 2002, "The GNU Awk User's Guide", GNU Project Web site, http://www.gnu.org/manual/gawk/index.html, date accessed July 16, 2002.

Jobjects, 2002, Jobject QuestAgent 5.0, http://www.jobjects.com/, date accessed July 16, 2002.

World Wide Web Consortium (W3C), the, 2002, the World Wide Web Consortium, http://www.w3c.org/, date accessed July 16, 2002.

Red Had, Inc. 2002, Cygwin, http://sources.redhat.com/cygwin/, date accessed July 16, 2002.

APPENDIX C. ACCOMPANYING CD-ROM AND OPERATING INSTRUCTIONS

System requirements for computer disks: IBM PC or 100% compatibles; Windows NT 4.0, Windows 95/98, Windows 2000 or higher; Internet Explorer 4.0 or higher; Acrobat reader 3.0 or higher.

The CD-ROM contains the Electronic Reference Library (ERL) project Web site, which is developed and maintained for improving knowledge management, communication and coordination for the ERL project. The Web site was created using MS FrontPage XP. The opening page is "index.htm". Because the Discussion Board of the Web site are to be run on a Web server, this feature is not available on this CD. To access the full featured Web site, please visit http://erl.cce.iastate.edu/ERL/.

REFERENCES

10 Gigabit Ethernet Alliance, 2002, "10 Gigabit Ethernet Technology Overview White Paper", http://www.10gea.org/Tech-whitepapers.htm, date accessed July 15, 2002.

3D/International, "Web-based Information Systems for Project Management", 3D/International, http://www.3di.com/, date accessed July 15, 2002.

Barnum, Carol M. 2002. *Usability Testing and Research*, New York, NY: Longman.

Dumas, Joseph S., and Janice C. Redish 1993. *A Practical Guide to Usability Testing*, Norwood, NJ: Ablex.

GNU Project, 2002, "The GNU Awk User's Guide", GNU Project Web site, http://www.gnu.org/manual/gawk/index.html, date accessed July 16, 2002

GSM Association, the 2002, "Introduction to 3G", GSM World, http://www.gsmworld.com/technology/3g/intro.shtml, date accessed July 15, 2002.

Microsoft Corporation, 2000, "Logging Site Activity", Microsoft Windows 2000 Server Documentation, http://www.microsoft.com/windows2000/en/server/iis/, date accessed July 15, 2002.

OpenWebScope, 2001, Web Statistics Software – OpenWebScope Homepage, http://www.openwebscope.com/, date accessed July 15, 2002.

Robert H'obbes' Zakon 2002, "Hobbes' Internet Timeline v 5.6", http://www.zakon.org/robert/internet/timeline/, date accessed July 15, 2002.

Lemay, Laura 1997, *Teach Yourself Web Publishing with HTML 3.2 in 14 Days*, Indianapolis, Indiana: Sams.net

Nielsen, Jacob 1993, *Usability Engineering*, Boston: Academic Press.

Stout, Rick 1997, "Web Site Stats: Tracking Hits and Analyzing Traffic", Berkeley, CA: Osborne McGraw-Hill.

Thorpe, T., and Mead, S. 2001. "Project-Specific Web Sites: Friend of Foe?" J. Construction Engineering and Management, ASCE, 127(5), 406-413.

Turban, E., Aronson, J. E. 2001. *Decision Support Systems and Intelligent Systems*, Upper Saddle River, New Jersey: Prentice Hall.

Varney, L. A. 1997, "Future of Integrated Information Management in Construction", Construction Congress V, p. 563 – 572, ASCE.

World Wide Web Consortium, the 1999, HTML 4.01 Specification, the World Wide Web Consortium Web site, http://www.w3.org/TR/html4/, date accessed July 15, 2002.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to Dr. Walters, for his expertise, instructions, patience and encouragement, which have been benefited me so much in the preparation of this thesis. I would also like to thank Dr. Jahren and Dr. Honeycutt, for their inspirational suggestions and instructions. My appreciation also goes to the preceding and current ERL team members either at Iowa State University or Iowa Department of Transportation, for the many benefits and helps that they have rendered me either through their writings or in person.

Secondly, I would like to thank the instructors of the courses I have taken, and the staff members of the Department of Civil and Construction Engineering, for their help in the past two years.

Last, I would give my unreserved gratitude to my family, for their unconditional support to me in my life.